T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY ARI KOLONİ ALGORİTMASININ SINIRLAMALI OPTİMİZASYON PROBLEMLERİ ÜZERİNDE PERFORMANS ANALİZİ

Hazırlayan
Demet ALICI KARACA

Danışman
Doç. Dr. Bahriye AKAY

Yüksek Lisans Tezi

Haziran 2019
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY ARI KOLONİ ALGORİTMASININ SINIRLAMALI OPTİMİZASYON PROBLEMLERİ ÜZERİNDE PERFORMANS ANALİZİ

(Yüksek Lisans Tezi)

Hazırlayan
Demet ALICI KARACA

Danışman
Doç. Dr. Bahriye AKAY

Haziran 2019
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Demet ALICI KARACA

İmza:

"Yapay Arı Koloni Algoritmasının Sınırlamalı Optimizasyon Problemleri Üzerinde Performans Analizi" adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan
Demet ALICI KARACA

Danışman
Doç. Dr. Bahriye AKAY

Bilgisayar Mühendisliği ABD Başkanı
Prof. Dr. Veysel ASLANTAŞ

Doç. Dr. Bahriye AKAY danışmanlığında **Demet ALICI KARACA** tarafından hazırlanan **"Yapay Arı Koloni Algoritmasının Sınırlamalı Optimizasyon Problemleri Üzerinde Performans Analizi"** adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği** Anabilim Dalında **Yüksek Lisans** tezi olarak kabul edilmiştir.

14/06/2019

JÜRİ:

Danışman : Doç. Dr. Bahriye AKAY .....

Üye : Prof. Dr. Derviş KARABOĞA .....

Üye : Prof. Dr. Zeki YETGİN .....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun 25/06/2019 tarih ve 2019/36-07 sayılı kararı ile onaylanmıştır.

25../.06../.2019.

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

ÖNSÖZ / TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, kıymetli bilgi, birikim ve tecrübeleri ile her zaman bana yol gösteren ve destek olan, kullandığı her kelime ile ufkumu açan değerli danışman hocam Doç. Dr. Bahriye Akay'a sonsuz teşekkür ve saygılarımı sunarım.

Değerli görüş ve önerileri için Prof. Dr. Derviş Karaboğa'ya ve Prof. Dr. Zeki Yetgin'e teşekkür ederim.

Bu süreçte her zaman yanımda olan aileme, anlayış ve desteğinden dolayı eşim Yunus Karaca'ya ve varlığıyla güç veren oğlum Miraç Efe'ye teşekkür ederim.

Demet ALICI KARACA

Haziran 2019, KAYSERİ

YAPAY ARI KOLONİ ALGORİTMASININ SINIRLAMALI OPTİMİZASYON PROBLEMLERİ ÜZERİNDE PERFORMANS ANALİZİ

Demet ALICI KARACA
Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Haziran 2019
Danışman : Doç. Dr. Bahriye AKAY

ÖZET

Yapay Arı Kolonisi (ABC) algoritması sürü zekası algoritmalarının arasında performansıyla öne çıkan algoritmalardan biridir. Temel ABC algoritmasının sınırlamasız optimizasyon problemlerinin çözümünde etkinliğinin görülmesiye tasarım parametrelerinin bazı koşullarla kısıtlandığı ve optimum değerin kabul edilebilir bölge içinde olması gerektiği sınırlamalı optimizasyon problemlerini çözmek için de ABC algoritmasının farklı versiyonları geliştirilmiştir. Problemlerde istenen kısıtları sağlamak amacıyla ceza terimine dayalı metotlar, çözümleri kabul edilebilir bölgede tutan metotlar, kabul edilebilir ve kabul edilebilir olmayan çözümler arasında ayrım yapan metotlar ve karma metotlar kullanılmıştır. Bu tez çalışmasında da sınırlamalı optimizasyon problemlerini çözmek amacıyla temel ABC algoritmasına sınırlama ele alış metotlarından ceza terimine dayalı metotlar içerisindeki ceza fonksiyonları (penalty function), bu ceza fonksiyonlarına farklı bir yaklaşım getiren rasgele sıralama (stochastic ranking) ve stokastik Deb kuralları entegre edilerek yeni yöntemler önerilmiştir. Geliştirilen yöntemler literatürde sıklıkla kullanılan sınırlamalı test problemleri üzerinde test edilmiş ve performansı literatürdeki rasgele sıralama (stochastic ranking), geliştirilmiş rasgele sıralama (improved stochastic ranking), aşırı ceza yaklaşımı (over-penalty approach), genetik algoritma, basit çok üyeli evrimsel strateji (simple multimembered evolution strategy), diferansiyel gelişim, parçacık sürüsü optimizasyonu algoritması ve ABC algoritması ile karşılaştırılmıştır. Yapılan analizler sonucunda tez kapsamında geliştirilen yöntemlerin belli parametre değerleriyle sınırlamalı optimizasyon problemlerinin çözümünde karşılaştırılan diğer algoritmalarla benzer ya da daha iyi sonuçlar elde edilmiştir.

Anahtar Kelimeler: Yapay arı kolonisi algoritması (ABC), sınırlamalı optimizasyon, sınırlama ele alış metotları.

PERFORMANCE ANALYSIS OF ARTIFICIAL BEE COLONY ALGORITHM ON CONSTRAINED OPTIMIZATION PROBLEMS

Demet ALICI KARACA
Erciyes University, Graduate School of Natural and Applied Sciences
M.Sc. Thesis, January 2019
Supervisor: Assoc. Prof. Bahriye AKAY

ABSTRACT

Artificial Bee Colony (ABC) algorithm is one of the algorithms that stands out with its performance among swarm intelligence algorithms. Since the effectiveness of the basic ABC algorithm in solving unconstrained optimization problems, different versions of the ABC algorithm have been developed in order to solve the constrained optimization problems where the design parameters are restricted by certain conditions and the optimum value should be within the feasible region. In order to solve the problems by taking into consideration the constraints, methods based on penalty functions, methods based on preserving feasibility of solutions, methods which distinguish between feasible and infeasible solutions and hybrid methods were used. In this thesis study, in order to solve the constrained optimization problems, new methods have been proposed by integrating penalty functions, stochastic ranking and stochastic Deb rules to the basic ABC algorithm. The developed methods have been tested on well-known constrained test problems in the literature and the results have been compared with other state-of-the-art algorithms, stochastic ranking, improved stochastic ranking, over-penalty approach, genetic algorithm, simple multimembered evolution strategy, differential evolution algorithm, particle swarm optimization algorithm, ABC algorithm. The overall results indicate that developed methods showed similar or better performance compared to the other algorithm to solve constrained optimization problems when certain parameter values are provided.

Keywords: Artificial bee colony algorithm (ABC), constrained optimization, constraint-handling.

İÇİNDEKİLER

YAPAY ARI KOLONİ ALGORİTMASININ SINIRLAMALI OPTİMİZASYON PROBLEMLERİ ÜZERİNDE PERFORMANS ANALİZİ

BİLİMSEL ETİĞE UYGUNLUK SAYFASI i

YÖNERGEYE UYGUNLUK SAYFASI ii

KABUL VE ONAY iii

ÖNSÖZ / TEŞEKKÜR iv

ÖZET v

ABSTRACT vi

İÇİNDEKİLER vii

TABLOLAR LİSTESİ x

ŞEKİLLER LİSTESİ xii

GİRİŞ **1**

1. BÖLÜM
OPTİMİZASYON

1.1. Optimizasyon Kavramı **4**

1.1.1. Veri ve Bilgi Toplama **5**

1.1.2. Problem Tanımı **5**

1.1.3. Tasarım Parametrelerinin Tanımı **5**

1.1.4. Optimizasyon Kriteri **6**

1.1.5. Sınırlamaların Formülizasyonu **7**

1.2. Optimizasyon Problemlerinin Sınıflandırılması **7**

1.2.1. Sınırlamaların Varlığına Göre Sınıflandırma **8**

1.2.2. Tasarım Parametrelerinin Yapısına Göre Sınıflandırma **8**

1.2.3. İçerdiği Denklemlerin Yapısına Göre Sınıflandırma **8**

1.2.4. Tasarım Parametrelerinin Alabileceği Değerlere Göre Sınıflandırma . . . 8
1.2.5. Değişkenlerin Deterministikliğine Göre Sınıflandırma . . . 9
1.2.6. Fonksiyonların Ayrılabilirliğine Göre Sınıflandırma . . . 9
1.2.7. Amaç Fonksiyonunun Sayısına Göre Sınıflandırma . . . 9
1.3. Optimizasyon Metotları . . . 9

2. BÖLÜM
SINIRLAMALI OPTİMİZASYON

2.1. Sınırlamalı Optimizasyona Giriş . . . 12
2.2. Analitik Metotlar . . . 13
2.2.1. Eşitlik Sınırlamalı Çok-değişkenli Optimizasyon . . . 14
2.2.2. Eşitsizlik Sınırlamalı Çok-Değişkenli Optimizasyon . . . 17
2.3. Sınırlamaları Ele Alış Metotları . . . 19
2.3.1. Ceza Fonksiyonları . . . 19
2.3.1.1. Stokastik Sıralama . . . 23
2.3.2. Deb Kurallarına Dayalı Seçim Stratejisi . . . 24
2.4. Sınırlamaları Ele Alış Metotlarına Dayalı Modern Sezgisel Algoritmalar ile Yapılan Çalışmalar . . . 25

3. BÖLÜM
YÖNTEM

3.1. Yöntem . . . 30
3.2. Arıların Doğal Yaşamı ve Yiyecek Arama Davranışları . . . 30
3.3. Yapay Arı Koloni Algoritması . . . 32
3.4. ABC Algoritmasının Sınırlamalı Optimizasyon Problemlerinin Çözümü İçin Uyarlanması . . . 36
3.4.1. Ceza Fonksiyonlarını Kullanarak ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü . . . 36
3.4.2. Stokastik Beslemeli ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü . . . 38
3.4.3. DEB-SR Yöntemini Kullanarak ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü . . . 39

4. BÖLÜM
BULGULAR

4.1. Deneysel Çalışmalar **41**

4.2. Test Problemleri **41**

4.3. Kontrol Parametreleri ve Değerleri **42**

4.4. İstatistiksel Sonuçlar **44**

4.4.1. PEN-ABC Sonuçları **44**

4.4.2. SR-ABC Sonuçları **51**

4.4.3. SR-ABC-DE Sonuçları **58**

4.4.4. DEB-SR-ABC Sonuçları **63**

4.4.5. Önerilen Yöntemlerin Kıyaslanması **65**

4.4.6. Literatürdeki Yöntemlerle Kıyaslama **66**

4.4.6.1. PEN-ABC'nin Kıyaslanması **66**

4.4.6.2. SR-ABC'nin Kıyaslanması **67**

4.4.6.3. DEB-SR-ABC'nin Kıyaslanması **68**

5. BÖLÜM
TARTIŞMA, SONUÇ ve ÖNERİLER

5.1. Tartışma, Sonuç ve Öneriler **73**

KAYNAKLAR **74**

EKLER **79**

1.1. Sınırlamalı Test Problemleri **80**

ÖZGEÇMİŞ **97**

TABLOLAR LİSTESİ

Tablo 1.1. Optimizasyon Metotları. 11

Tablo 4.1. Test problemlerinin temel özelliklerinin özeti 42

Tablo 4.2. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g01-g12 test problemleri için istatistiksel sonuçları 47

Tablo 4.3. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g13-g24 test problemleri için istatistiksel sonuçları 48

Tablo 4.4. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler 49

Tablo 4.5. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler 50

Tablo 4.6. Stokastik beslemeli ABC algoritmasının g01-g12 test problemleri için istatistiksel sonuçları 54

Tablo 4.7. Stokastik beslemeli ABC algoritmasının g13-g24 test problemleri için istatistiksel sonuçları 55

Tablo 4.8. Stokastik beslemeli ABC algoritmasının g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler 56

Tablo 4.9. Stokastik beslemeli ABC algoritmasının g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler 57

Tablo 4.10. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğnii kullanarak g01-g12 test problemleri için istatistiksel sonuçları 60

Tablo 4.11. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğini kullanarak g13-g24 test problemleri için istatistiksel sonuçları 61

Tablo 4.12. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğnii kullanarak g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler . . 62

Tablo 4.13. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğini kullanarak g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler . . 63

Tablo 4.14. DEB-SR yöntemiyle ABC algoritmasının 24 test problemi için istatistiksel sonuçları 64

Tablo 4.15. DEB-SR yöntemiyle ABC algoritmasının 24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler 65

Tablo 4.16. SR, ISR, OPA, GA, SMES, PSO, DE, ABC, PEN-ABC, SR-ABC ve DEB-SR-ABC algoritmalarının 13 test fonksiyonu için elde edilen sonuçların en iyi değerleri. (-) kabul edilebilir çözümün bulunamadığı anlamına gelir. 70

Tablo 4.17. SR, ISR, OPA, GA, SMES, PSO, DE, ABC, PEN-ABC, SR-ABC ve DEB-SR-ABC algoritmalarının 13 test fonksiyonu için elde edilen sonuçların ortalama değerleri. (-) kabul edilebilir çözümün bulunamadığı anlamına gelir. 71

Tablo 4.18. DEB-SR-ABC algoritmasının diğer algoritmalarla karşılaştırıldığında başarı oranı. + algoritmanın daha iyi, - daha kötü olduğu anlamına gelir. Benzer performans gösterenlerin her ikisi de + alınmıştır. 72

ŞEKİLLER LİSTESİ

Şekil 1.1. $f(x)$ fonksiyonunun minimumu ($-f(x)$ fonksiyonunun maksimumu) 4

Şekil 3.1. Yiyecek Arama Çevrimi 32

GİRİŞ

Optimizasyon kavramıyla günlük yaşantıda çok sık karşılaşılmaktadır. En kısa yol, minimum maliyet, maksimum kar, en iyi tasarım, asgari hata, en iyi yönetim gibi ifadelerle optimizasyon farklı alanlarda duyulmaktadır. Bu ifadeler işletme, ekonomi, tasarım, üretim, matematik ve mühendislik gibi her alanı ilgilendirmektedir. Optimizasyon verilen koşullar altında mümkün olan en iyi sonuca ulaşma eylemidir.

Optimizasyon sürecinde problemle ilgili bilgi toplandıktan sonra ihtiyaçlar belirlenir ve problem tanımlanır. Sistemi tanımlayan tasarım değişkenlerine (parametreler) karar verilir ve bir çözümün diğerine göre iyiliğini belirleyecek olan tasarım değişkenlerine bağlı olarak en küçük veya en büyük yapılacak olan amaç fonksiyonu oluşturulur. Optimizasyon amaç fonksiyonunu en iyileyen tasarım değişkenlerini bulmaktır. Sınırlamaların varlığı, tasarım parametrelerinin yapısı, amaç fonksiyonunun içerdiği denklem yapısına göre farklılık gösteren pek çok optimizasyon problemi türü vardır. Bu farklı türdeki problemleri çözmek için ise analitik ve modern optimizasyon metotları geliştirilmiştir. Hesaplama metotları, lineer programlama, geometrik programlara gibi analitik yöntemlerin zor problemlerde yetersiz kalmasından dolayı Genetik algoritma, Diferansiyel Gelişim algoritması, Yapay Arı Kolonisi algoritması gibi modern (sezgisel) yöntemlere başvurulmuştur. Bu algoritmalar ilk olarak sınırlamasız optimizasyon problemlerinin çözümü için önerilseler de sınırlama ele alış metotlarının (constraint handling) entegre edilmesi ile sınırlamalı optimizasyon problemlerini de çözebilir hale gelmektedir.

Yapay Arı Koloni algoritması arıların yiyecek arama davranışını simüle eden modern optimizasyon algoritmalarındandır. Sınırlamasız problemlerin çözümündeki başarısıyla literatürdeki diğer modern algoritmalar arasında öne çıkmaktadır. Etkin performansı göz önüne alındığında Yapay Arı Koloni algoritmasına sınırlama ele alış yöntemleri entegre edilerek sınırlamalı problemleri çözebilir hale getirilmiştir. Sınırlama ele alış

yöntemlerinden sınırlamayı ihlal eden çözümlerin cezalandırılması mantığına dayanan ceza fonksiyonları (penalty function), ceza fonksiyonlarına farkli bir yaklaşım getiren stokastik sıralama (stochastic ranking) [16] ve algoritmanın seleksiyon aşamasında kullanılan Deb'in kuralları [17] farklı bir şekilde kullanılarak Yapay Arı Koloni algoritmasının sınırlamalı optimizasyon problemleri üzerindeki performansı analiz edilmiştir.

Tezin Amacı

Bu tez çalışmasında kullanılan ceza fonksiyonları Runnarson ve Yao [50] tarafından gelişim stratejisi (Evolution Strategy) algoritmasında, stokastik sıralama yaklaşımı Runnarson ve Yao [16] tarafından gelişim stratejisi algoritmasında ve Deb'in kuralları [40] Yapay Arı Koloni algoritmasının seleksiyon aşamasında sınırlamalı optimizasyon problemlerini çözmek amacıyla kullanılmış ve başarılı sonuçlar elde edilmiştir. Bu çalışmalardan yola çıkarak sınırlamalı optimizasyon problemlerinin çözümüne yönelik Yapay Arı Koloni algoritması kullanılarak yeni yöntemler geliştirmek ve daha etkin sonuçlar üretmek amaçlanmıştır. Sınırlama ele alış metodunun iyiliği algoritmaya göre değişebileceğinden hangi sınırlama ele alış metodunun ABC algoritmasının yapısına uygun olduğunun incelenmesi hedeflenmiştir. Yapay Arı Koloni algoritmasına ceza fonksiyonları, rasgele sıralama ve stokastik Deb kuralları entegre edilerek problemlere uygulanmış ve performans karşılaştırması literatürde mevcut olan sınırlamalı test problemleri üzerinden yapılmıştır.

Tezin Organizasyonu

Yapay Arı Koloni (ABC) algoritmasının sınırlamalı optimizasyon problemleri üzerinde performans analizinin yapıldığı bu tez çalışmasının organizasyonu aşağıda belirtildiği gibidir:

Birinci bölümde, optimizasyon kavramı anlatılarak konuya giriş yapılmıştır. Optimizasyon problem türlerinin nasıl sınıflandırıldığı açıklanmış ve bu problemlere getirilen çözüm metotlarına değinilmiştir.

2. Bölümde, tez konusunun çıkış noktası olan sınırlamalı optimizasyon problemi açıklanmış ve sınırlamalı optimizasyon probleminin çözmek için kullanılan sınırlamaları

ele alış metotlarına değinilmiştir. Bu metotlardan tez çalışmasında kullanılacaklar detaylı bir şekilde anlatılmıştır. Ayrıca sınırlamalı optimizasyon problemini çözmek amacıyla yapılan çalışmalar sunulmuştur.

3. Bölümde öncelikle Yapay Arı Koloni (ABC) algoritması hakkında kısa bilgi verilmiştir. Önceki bölümde anlatılan sınırlama ele alış metotları sınırlamalı optimizasyon problemlerine çözüm getirmek amacıyla temel ABC algoritmasına entegre edilmiştir.

4. Bölümde tez çalışmasında elde edilen bulgular tablolar halinde gösterilmiş, literatürdeki algoritmalarla karşılaştırılmış ve sonuçlar yorumlanmıştır.

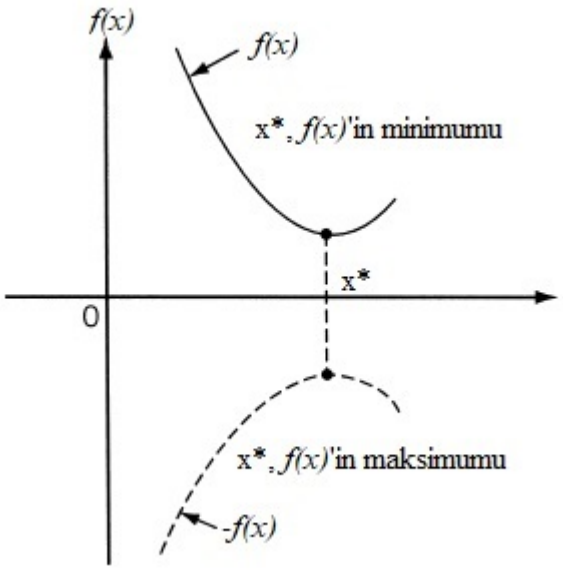
5. Bölümde ise sonuçlara yer verilmiştir.

1. BÖLÜM

OPTİMİZASYON

1.1. Optimizasyon Kavramı

Optimizasyon, istenen koşullar altında bir problemin minimum ya da maksimum değerini veren parametre değerlerini bulma işlemidir. Amaç maliyeti minimuma indirmek ve sonuç olarak faydayı maksimuma çıkarmaktır. Bu ifadenin matematiksel tanımı ise şu şekildedir: S araştırma uzayı, F, $F \subseteq S$ şeklinde kabul edilebilir çözümler bölgesi, $\vec{x}$ tasarım değişkenleri vektörü ve f amaç fonksiyonu olsun. Optimizasyon, her $\vec{y} \in F$ için minimizasyon problemlerinde$f(\vec{x}) \leq f(\vec{y})$'yi sağlayan $\vec{x} \in F$ çözümünün bulunması işlemidir. Bu durumda $\vec{x}$ küresel minimum olmaktadır. Amaç fonksiyonu f, nümerik $(f : S \to R)$ ya da kombinasyonel $((f : SxS \to R)$ olabilmektedir [1].



Şekil 1.1. $f(x)$ fonksiyonunun minimumu ($-f(x)$ fonksiyonunun maksimumu)

Optimizasyon işleminde etkin sonuç alabilmek için çözülmek istenen problemin doğru bir

şekilde analiz edilerek modellenmesi gerekir. Problemin tanımlanması ve matematiksel olarak formülize edilmesi modelleme olarak adlandırılır. Problem için oluşturulan modele göre, en iyi çözümü üreten tasarım değişkenlerini bulma işlemi optimizasyondur.

Optimum tasarım problemlerinin modellenmesi oldukça önemlidir ve beş adımdan oluşur [2].

1. Veri ve Bilgi Toplama
2. Problemin Tanımı
3. Tasarım Parametrelerinin Tanımı
4. Optimizasyon Kriteri
5. Sınırlamaların Formülasyonu

1.1.1. Veri ve Bilgi Toplama

Modelleme işlemi problem ve ihtiyaçlarla ilgili bilgi toplamakla başlar. Malzeme özellikleri, kaynak limitleri, malzemelerin maliyeti, performans gereklilikleri gibi başka bilgiler de gerekebilir. Bu nedenle problemle alakalı tüm bilgiler toplanmalıdır.

1.1.2. Problem Tanımı

Problemin tanımlayıcı ifadesini oluşturmakla devam edilir. Oluşturulan ifade problemin tüm amaçlarını ve karşılaması gereken gereksinimleri tanımlar. Bazı problemlerde problemin tanımı belirsiz olabilir. Bu tür problemleri formülize etmek ve çözmek için problemin modeli hakkında varsayımlar yapılır.

1.1.3. Tasarım Parametrelerinin Tanımı

Formülasyon sürecinde bu adımda tasarım değişkenleri olarak adlandırılan sistemi tanımlayan değişkenler seti tanımlanır. Genellikle bu değişkenler optimizasyon değişkenleri olarak da adlandırılır. Değişkenler bağımsız olmalıdır ve istenilen her

değer atanabilmelidir. Çünkü atanan her farklı değer farklı tasarımlar üretir ve bağımsız parametrelerin sayısı problem için *serbestlik derecesini* verir.

Problem için uygun tasarım parametreleri seçilmezse formülizasyon ya yanlış olacaktır ya da oluşturulması mümkün olmayacaktır. Bu nedenle ilk aşamada tasarım parametrelerinin belirlenmesi için tüm seçenekler araştırılmalıdır. Bazı durumlarda belirlenen serbestlik derecesinden daha fazla tasarım parametresi atanmak istenebilir. Bu formülizasyon için esneklik katar. Fazladan atanan değişkenlere sabit değer verilerek o değişken formülizasyondan elenebilir.

Tasarım parametrelerinin belirlenmesinin oldukça zor olduğu problemler de olabilir. Böyle bir durumda tüm değişkenlerin listesi hazırlanarak her değişken ayrı ayrı göz önüne alınır ve onun tasarım parametresi olarak uygun olup olmadığına karar verilir. Eğer geçerli bir tasarım parametresi olduğuna karar verilirse bir başlangıç çözümü seçmek için o değişkene nümerik bir değer verilebilmelidir.

Tasarım parametreleri $\vec{x}$ vektörü ile temsil edilir. Özetlersek; bir problem için tasarım parametrelerini belirlerken aşağıdakiler göz önüne alınmalıdır:

- Optimizasyon problemini uygun şekilde formülize etmek için gereken tasarım parametreleri minimum sayıda olmalıdır. Yani serbestlik derecesi küçük olmalıdır.
- Tasarım parametreleri mümkün olduğu kadar birbirlerinden bağımsız olmalıdır. Değilse bile aralarında bazı eşitlik sınırlamaları olmalıdır.
- Çoğu bağımsız parametrede olduğu gibi problemin formülizasyon aşamasında tasarım parametrelerine mümkün olduğunca değer atanabilir olmalıdır.
- Sistemin başlangıç çözümü belirlenirken tanımlanan her tasarım parametresine nümerik değer verilmelidir.

1.1.4. Optimizasyon Kriteri

Sistem için bir çok kabul edilebilir tasarım olabilir. Ancak bunlardan bazıları diğerlerinden daha iyidir. Bu durumda tasarımların nasıl karşılaştırıldığına ve hangisinin daha iyi olduğuna karar verileceği belirlenmelidir. Bunun için tasarımın başarısını

belirleyen kriterler olmalıdır. Optimizasyon kriteri nümerik bir değerin elde edilebildiği skaler bir fonksiyon olmalıdır. Yani $\vec{x}$ tasarım parametleri vektörünün bir fonksiyonu olmalıdır. Bu kriter optimum tasarım problemlerinde *amaç fonksiyonu* olarak adlandırılır. Bu fonksiyon ihtiyaca göre maksimize ya da minimize edilir. Minimize edilmeye çalışılan optimizasyon kriteri literatürde genellikle *maliyet fonksiyonu* olarak geçer.

Uygun amaç fonksiyonunun seçimi tasarım problemlerinde önemli bir karardır. Açık ve net bir amaç fonksiyonu tanımlanmalıdır. Bazı durumlarda ise iki ve ya daha fazla amaç fonksiyonu tanımlanabilir. Bunlar *çok amaçlı optimizasyon problemi* olarak tanımlanır.

1.1.5. Sınırlamaların Formülizasyonu

Modelleme sürecindeki son adım tüm sınırlamaları tanımlamak ve onlar için ifadeler oluşturmaktır. Tasarımın araştırılacağı uzayı sınırlandıran bütün ifadeler *sınırlamalar* olarak adlandırılır. Uygun sınırlamaların tanımlanması ile gerçeğe en yakın sistem tasarlanmalı, mevcut kaynakları kullanabilmeli ve performans gereksinimlerini karşılamalıdır.

Sınırlamalar tasarım parametrelerine bağlı olmalıdır. Çünkü tasarım parametrelerinin farklı değerleri için sınırlama değerleri de değişir. Bundan dolayı anlamlı bir sınırlama fonksiyonu, tasarım parametrelerinden en az birinin bir fonksiyonu olmalıdır.

Modelleme işlemi bittikten sonra problemin çözümüne geçilir. Sınırlamaların olup olmaması, tasarım parametrelerinin yapısı, problemin yapısı, problemin içerdiği denklemlerin yapısı vs. incelenerek probleme uygun çözüm metodu kullanılır. Problemleri çözmek için kullanılan bu metotlar optimizasyon metotları olarak tanımlanır.

1.2. Optimizasyon Problemlerinin Sınıflandırılması

Literatürde çok sayıda ve farklı türde optimizasyon problemi bulunmaktadır ve tek bir metot bütün optimizasyon problemlerinin çözümü için yeterli değildir. Bu nedenle farklı türdeki optimizasyon problemlerinin çözümü için çok sayıda metot geliştirilmiştir [3]. Öncelikle problem türleri ve bunlar üzerinden çözüm metotlarına değinilecektir.

1.2.1. Sınırlamaların Varlığına Göre Sınıflandırma

Optimizasyon problemleri sınırlamaların olup olmamasına göre *sınırlamalı* yada *sınırlamasız* optimizasyon problemi olarak adlandırılır.

1.2.2. Tasarım Parametrelerinin Yapısına Göre Sınıflandırma

Tasarım parametrelerinin yapısına bağlı olarak optimizasyon problemleri iki alt kategoriye ayrılabilir. İlk kategoride tasarım parametrelerinin zamana bağlı olarak değişmediği yada belirsizlik içermediği problem türleri vardır. Bu problemlere *statik optimizasyon problemleri* denir. İkinci kategoride ise tasarım parametreleri zamana bağlı değişkenlik göstermektedir. Bu problem türünde ise amaç, zamana ya da olaylara bağlı değişiklik gösteren optimal çözümü bulmaktır. Bu tarz problemlere *dinamik optimizasyon problemi* denir.

1.2.3. İçerdiği Denklemlerin Yapısına Göre Sınıflandırma

Optimizasyon problemleri için başka önemli bir sınıflandırma amaç fonksiyonu ve sınırlamalar için kullanılan ifadelerin yapısıdır. Bu sınıflandırmaya göre optimizasyon problemi lineer, nonlineer, geometrik veya kuadratik olabilir. Amaç fonksiyonu ve tüm sınırlamalar doğrusal (lineer) ise *Doğrusal Programlama*, amaç ve sınırlama fonksiyonlarından herhangi biri doğrusal değil (nonlineer) ise *Doğrusal Olmayan Programlama (NLP)*, amaç fonksiyonu ve sınırlama fonkiyonları, c_k ve x_i pozitif reel sayılar, a_ik reel sayı olmak üzere $f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} ... x_n^{a_{nk}}$ şeklinde yani posinomial ise *Geometrik Programlama (GMP)* adını alır. Kuadratik amaç fonksiyonu ve lineer sınırlamalı nonlineer programlama problemi ise *Kuadratik Programlama* olarak tanımlanır..

1.2.4. Tasarım Parametrelerinin Alabileceği Değerlere Göre Sınıflandırma

Tasarım parametrelerinin izin verilen değerlerine göre, optimizasyon problemleri tamsayı ve reel değerli programlama problemleri olarak sınıflandırılabilir. Bir optimizasyon probleminin tüm yada bazı tasarım parametreleri sadece tamsayı (veya

ayrık (discrete)) değer almakla sınırlandırılırsa, problem *tamsayı programlama problemi* olarak adlandırılır. Diğer yandan, tüm tasarım parametrelerinin reel değer almasına izin verilirse, optimizasyon problemi *reel değerli programlama problemi* olur. Başka bir grup ise sürekli ve ayrık optimizasyon problemidir. Tasarım parametrelerinin alacağı değerler sürekli değerler ise bu tür problemler *sürekli optimizasyon problemi* olarak adlandırılır. Ayrık niceliklerin optimal olarak düzenlenmesi, gruplanması, sıralanması veya seçilmesi problemi ise *ayrık optimizasyon problemi (kombinatoryal)* dir.

1.2.5. Değişkenlerin Deterministikliğine Göre Sınıflandırma

İçerdiği değişkenlerin deterministikliğine göre, optimizasyon problemleri deterministik ve stokastik programlama problemleri olarak sınıflandırılır. Problemdeki parametrelerin tümü yada bazıları olasılıksal (nondeterministik veya stokastik) olduğunda *stokastik programlama problemi* olarak kabul edilir. Aksi durumda ise *deterministik programlama problemi* olarak isimlendirilir.

1.2.6. Fonksiyonların Ayrılabilirliğine Göre Sınıflandırma

Optimizasyon problemleri amaç ve sınırlama fonksiyonlarının ayrılabilirliğine dayanarak sınıflandırılabilir. Eğer $f(x)$ fonksiyonu n tek-değişkenli fonksiyonun, $f_1(x_1), f_2(x_2), ...f_n(x_n)$, toplamı yani $f(x) = \sum_{i=1}^{n} f_i(x_i)$ şeklinde ifade edilebilirse *ayrılabilir (separable)* olduğu söylenir. Bu formda ifade edilemiyorsa *ayrılamayan (non-separable)* olarak isimlendirilir.

1.2.7. Amaç Fonksiyonunun Sayısına Göre Sınıflandırma

Minimize edilmesi gereken amaç fonksiyonlarının sayısına göre, optimizasyon problemleri *tek* ve *çok amaçlı programlama problemleri* olarak gruplandırılabilir.

1.3. Optimizasyon Metotları

Optimizasyon metotlarının çıkışı Newton, Lagrange ve Cauchy'nin çalışmalarına dayanır. Newton ve Leibnitz'in hesaplamaya olan katkıları sayesinde diferansiyel hesaplama

metotları geliştirilmiştir. Bernoulli, Euler, Lagrange ve Weirstrass tarafından varyasyon hesaplarının bulunmasıyla fonksiyonların minimumları bulunmuştur. Bilinmeyen yeni çarpanları içeren sınırlamalı optimizasyon problemler için optimizasyon metodu Lagrange adıyla anılmaya başlamıştır. Cauchy sınırlamasız minimizasyon problemleri için en dik iniş metodunun ilk uygulamasını yapmıştır. *Analitik optimizasyon metotlarında* birkaç değişkenli sınırlamasız fonksiyonun minimum yada maksimumunu bulmak için diferansiyel hesaplamalar kullanılabilir. Bu metotlar tasarım parametrelerinin iki kez türevlenebilir olduğunu ve türevin sürekli olduğunu varsayar. Eşitlik sınırlamalı optimizasyon problemleri için Lagrange Çarpanları yöntemi kullanılabilir. Problem eşitsizlik sınırlamalarına sahipse Kuhn-Tucker koşulları kullanılabilir. Ancak bu metotlar çözülmesi zor olabilen nonlineer eş-zamanlı denklemlere neden olabilir. Bu nedenle farklı türdeki problemlerin çözümü için analitik metotlara yakınsayan nümerik metotlar ortaya çıkmıştır. Nonlineer, lineer, geometrik, kuadratik ve tamsayı programlama metotları, metodun adında belirtilen problemlerin belirli gruplarının çözümü için kullanılabilir. Bu metotların çoğu bir başlangıç çözümünden başlayarak iteratif bir şekilde devam ederek yaklaşık çözümün bulunduğu nümerik yöntemlerdir. Bunlardan nonlineer programlama metodu her optimizasyon probleminin çözümünde kullanılabilen en genel yöntemdir.

Geleneksel matematiksel programlama metotlarından farklı olarak fonksiyonların türevlenebilir olmasını gerektirmeyen, problemin türünden bağımsız bazı optimizasyon metotları geliştirilmiştir. Bu metotlar *modern* ve ya *geleneksel olmayan optimizasyon metotları* olarak tanımlanır. Bu metotların çoğu biyolojik, moleküler, böcek sürüsü, nörobiyolojik sistemlerin belirli özellik ve davranışlarına dayanmaktadır. Genetik Algoritma (GA), Isıl İşlem, Parçacık Sürüsü Optimizasyonu (PSO), Karınca Koloni Optimizasyonu, Diferansiyel Gelişim (DE) Algoritması, Yapay Arı Kolonisi (ABC) Algoritması modern yöntemlerdendir [3, 4]. Bu yöntemlere karmaşık mühendislik problemlerinin çözümü için son yıllarda daha sık başvurulmaktadır. Genetik algoritmalar doğal genetik ve seleksiyon prensiplerine dayanmaktadır. Isıl İşlem çıkabileceği maksimum sıcaklığa kadar ısıtılan katıların soğutulmasını simüle eder. Hem Genetik Algoritma hem de Isıl İşlem yüksek olasılıkla global minimumu veya optimuma yakın bir değeri bulabilen stokastik metotlardır. Dolayısıyla ayrık optimizasyon problemlerinin çözümüne uygundur. Parçacık Sürü Optimizasyonu böcek, kuş ve ya balık gibi

canlıların koloni davranısına dayanır. Karınca Koloni Optimizasyonu yuvalarından yemek kaynağına en yakın yolu bulan karıncaların ortak davranışına dayanır. Diferansiyel Gelişim algoritması, sürekli uzayda optimizasyon problemlerini çözmek için geliştirilmiş etkili ve güçlü stokastik araştırma metodudur [5]. Yapay Arı Kolonisi Algoritması bal arılarının zeki yiyecek arama davranışını simüle eder [6]. Optimizasyon metotları genel haliyle Tablo 1.1'de verilmiştir [3].

Tablo 1.1. Optimizasyon Metotları.

Optimizasyon Metotları	
Analitik Metotlar	**Modern Metotlar**
Hesaplama Metotları	Genetik Algoritmalar
Varyasyon Hesaplamaları	Isıl İşlem
Nonlineer Programlama	Karınca Koloni Optimizasyonu
Geometrik Programlama	Parçacık Sürü Optimizasyonu
Kuadratik Programlama	Diferansiyel Gelişim Algoritması
Lineer Programlama	Yapay Arı Kolonisi Algoritması
Dinamik Programlama	
Tamsayı Programlama	
Stokastik Programlama	
Ayrılabilir Programlama	
Çok Amaçlı Programlama	

2. BÖLÜM

SINIRLAMALI OPTİMİZASYON

Optimizasyon metotları ilk olarak sınırlamasız optimizasyon problemlerinin çözümü için geliştirilmiştir. Ancak gerçek dünya problemlerinde çözümler belli sınırlamalara tabi olduğu için bu tür problemlerin çözümü için sınırlamalı optimizasyon metotları ortaya çıkmıştır. Bu bölümde, optimizasyon problemlerinden sınırlamalı optimizasyon problemi (SOP) detaylı bir şekilde anlatılacaktır. SOP'nin çözümünde kullanılan sınırlama ele alış metotları açıklanacak ve şimdiye kadar yapılan çalışmalara değinilecektir.

2.1. Sınırlamalı Optimizasyona Giriş

Sınırlamalı optimizasyon problemleri kaynaklarla ilgili kısıtların bulunduğu yapısal tasarım, ekonomi, yerleşim, lokasyon, VLSI mühendislik problemleri gibi gerçel problemlerde karşımıza çıkar [7]. Sınırlamalar arasındaki etkileşim, sınırlamalar ve amaç fonksiyonu arasındaki ilişki, bellek hacmi, hesaplama maliyeti ve karar değişkenleri üzerindeki kısıtlar sınırlamalı problemlerin çözümünü zorlaştırır [8].

Doğrusal olmayan (nonlineer) sınırlamalı optimizasyon problemi, eşitsizlik ve/veya eşitlik sınırlamalarına tabi olarak $f(\vec{x})$ amaç fonksiyonunu minimize eden $\vec{x}$ parametre vektörünü bulma işlemi olarak tanımlanır. Sürekli uzayda Eşitlik 2.1'de gösterildiği gibi formülize edilebilir.

$$\begin{aligned} & Minimize\ f(\vec{x}) \\ & g_j(\vec{x}) \leq 0,\ j = 1, ..., m \\ & h_k(\vec{x}) = 0,\ k = 1, ..., n \\ & x_i^l \leq x_i \leq x_i^u,\ i = 1, ..., D \end{aligned} \tag{2.1}$$

$\vec{x} = (x_1, x_2, x_3, ..., x_D)$ D-boyutlu tasarım parametreleri vektörü, $f(\vec{x})$ minimize

yada maksimize edilecek amaç fonksiyonu, m eşitsizlik sınırlamaları sayısı ve n eşitlik sınırlamaları sayısıdır. $g_j(\vec{x})$ ve $h_k(\vec{x})$ sırasıyla eşitsizlik ve eşitlik sınırlama fonksiyonlarıdır. x_i^l ve x_i^u, x_i parametresinin alt ve üst sınırlarıdır ve optimizasyon probleminin araştırma uzayını tanımlar. Eşitsizlik ve eşitlik sınırlamaları ise araştırma uzayında kabul edilebilir bölgeyi tanımlar [9]. Eşitlik sınırlamaları genellikle optimizasyon hesaplama işlemlerinde tam olarak karşılanmaz. Eşitlik sınırlamaları küçük bir ihlal değeri ε ile eşitsizlik sınırlamasına çevrilir. $g_j(x) \leq 0$ ve $|h_k(x) - \varepsilon| \leq 0$ ise $\vec{x}$ çözümü kabul edilebilir sayılır. Sonuçların kalitesi ε ihlal değerinin seçimine bağlıdır. Eğer çok küçük seçilirse algoritma kabul edilebilir çözümler bulamayabilir. Çok büyük seçilirse sonuçlar kabul edilebilir bölgeden uzaklaşabilir [10].

2.2. Analitik Metotlar

Sınırlamalı optimizasyon problemlerini çözmek için kullanılan bazı analitik optimizasyon metotları sürekli ve türevlenebilir fonksiyonların optimum çözümünü bulmakta kullanılabilir. Bu metotlar optimum noktayı bulmak için türeve dayalı teknikleri kullanır. Gerçek hayattaki problemlerin amaç fonksiyonu her zaman sürekli yada türevlenebilir olmadığı için analitik metotların gerçek hayat problemlerinde kullanımı sınırlıdır. Sınırlamalı optimizasyon problemlerinde sınırlamaların türüne göre çözüm metodu geliştirilmiştir.

Optimizasyon problemlerinde amaç, minimizasyon problemi için kabul edilebilir bölge içinde amaç fonksiyonunun minimum değerini veren optimum noktayı bulmaktır. Bu optimum nokta ise küresel minimum ile tanımlanır. x^* noktasında fonksiyonun değeri, F kabul edilebilir çözümler bölgesindeki her x noktasından daha küçük yada eşit değer alıyorsa x^* noktası *küresel minimum*dur. Kısaca F kabul edilebilir çözümler bölgesinde tüm x değerleri için $f(x^*) \leq f(x)$'dir. Eğer bu eşitsizlik yine F kabul edilebilir çözümler bölgesindeki x^*'in dar komşuluğundaki (N) tüm x noktalarında sağlanıyorsa x^*'da *yerel minimum*a sahiptir denir. x^* noktasının dar komşuluğu (N), δ küçük bir değer olmak üzere $N = \{x|x \in F \; ve \; \|x - x^*\| < \delta\}$ ile tanımlanır. Bu ifade geometrik olarak x^* noktası etrafındaki küçük kabul edilebilir bölgedir [2].

Optimizasyon problemlerinde elde edilen sonucun optimum nokta olup olmadığına karar

vermek için gerek ve yeter şartlar vardır.

Gerek Şart: $f(x)$ fonksiyonu $x = x^*$ noktasında maksimum yada minimuma sahipse x^*'da $f(x)$'in birinci dereceden türevi sıfıra eşittir.

$$\frac{\partial f}{\partial x_1}(x^*) = \frac{\partial f}{\partial x_2}(x^*) = ... = \frac{\partial f}{\partial x_n}(x^*) = 0 \tag{2.2}$$

Yeter Şart: x^* noktasının ekstrem nokta olması için yeter şart $f''(x^*)$ matrisinin (Hessian matris) pozitif tanımlı olmasıdır ya da x^* yerel maksimum olduğunda negatif tanımlı olmasıdır.

2.2.1. Eşitlik Sınırlamalı Çok-değişkenli Optimizasyon

$$minimize\ f(\vec{x}) \tag{2.3}$$

$$g_j(x) = 0,\ j = 1, ..., m \tag{2.4}$$

$$\vec{x} = \begin{Bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{Bmatrix} \tag{2.5}$$

Burada optimum çözüm elde etmek için n değişken sayısı ve m sınırlama sayısı olmak üzere $m \leq n$ olmalıdır. Aksi taktirde sınırlamaların fazlalığından dolayı çözüm olmayacaktır. $m > n$ olduğunda problem çok fazla sayıda sınırlamalar içereceğinden dolayı aşırı tanımlanmış (overdefined) olur. Eşitlik sınırlamalı optimizasyon problemlerinin çözümü için birkaç metot vardır. Bunlar *doğrudan yerine koyma (direct substitution)*, *sınırlı varyasyon (constrained variation)* ve *Lagrange çarpanları (Lagrange multipliers)* metotlarıdır [3].

- **Doğrudan Yerine Koyarak Çözüm:** Doğrudan yerine koyma metodu ile n değişkenli ve m sınırlamalı bir problemde m eşitlik sınırlamalarını aynı anda çözmek mümkündür. $n - m$ değişkenin dışında kalan değişkenler, eşitlik sınırlamalarında yerine koyularak sınırlamalı problem sınırlamasıza çevrilir ve o şekilde çözümü yapılır. Bu metot teorik olarak basit bir metot olsa da pratikte problemlerin çözümü için pek de uygun değildir. Çünkü zor problemlerde sınırlamalar nonlineer olduğu için, bu kısıtlardan n-m değişkeni m cinsinden yazmak mümkün olmayabilir [3].

- **Sınırlamalı Varyasyon Metodu ile Çözüm:** Sınırlı varyasyondaki ana fikir sınırlamaları sağlayan $g_j(\vec{x}) = 0,\ j = 1, 2, ..., m$ her x noktasında $f(\vec{x})$'in birinci mertebeden türevinin kapalı form ifadesini bulmaktır. İstenen optimum noktalar türevin (df) sıfıra eşit olduğu noktalarda elde edilir [3].

- **Lagrange Çarpanları Metoduyla Çözüm:** Sınırlamaları eşitlik halinde olan problemlerde kullanılır. Lagrange metodu, optimum çözüm x^*'in düzenli (regular) nokta olması üzerine kuruludur. Bir eşitsizlik sınırlaması $g_j(x) \leq 0$ eşitliği sağlayan x^* noktasında $g_j(x^*) = 0$ ile sağlanıyorsa *aktif sınırlama*dır. Kabul edilebilir çözümler için eşitsizlik sınırlamaları aktif yada inaktif olabilir. Ancak eşitlik sınırlamaları kabul edilebilir tüm çözümler için aktiftir. Sınırlamaların sağlanmadığı durum ise *ihlal* olarak ifade edilir [2, 3].

  Düzenli (regular) nokta: x^* optimum noktasında eşitlik sınırlamaları sıfıra eşit (aktif sınırlama) ve bu sınırlamaların birinci türev (gradyant) vektörleri birbirinden lineer olarak bağımsızdır. Lineer bağımsızlık, iki vektörün gradyantlarının birbirine paralel olmaması ve herhangi bir gradyant vektörünün bir başkasının lineer fonksiyonu olarak yazılamamasıdır [2].

  Eşitlik 2.3 ve Eşitlik 2.4'te tanımlanan optimizasyon problemi için x^* yerel minimuma sahip düzenli bir noktadır. $\lambda_j^*,\ j = 1, ..., m$, Lagrange çarpanı olmak üzere n değişkenli m sınırlamalı genel bir problem için Lagrange çarpanı her bir sınırlamayla çarpılıp amaç fonksiyonuna eklenir. Bu fonksiyona *Lagrange fonksiyonu* denir. Eşitlik 2.6 ile Lagrange fonksiyonu oluşturulmuştur.

$$L(x, \lambda) = f(x) + \sum_{k=1}^{m} \upsilon_k^* g_k(x) = f(x) + \lambda^T g(x) \tag{2.6}$$

  $n + m$ bilinmeyenli bir fonksiyon haline gelen Lagrange fonksiyonunun ekstremum noktası aynı zamanda f amaç fonksiyonunun da çözümüne karşılık gelir. Bu nokta için gerek şart Eşitlik 2.7'de verilmiştir.

$$\begin{aligned} &\tfrac{\partial L}{\partial x_i} = \tfrac{\partial f}{\partial x_i} + \textstyle\sum_{j=1}^{m} \lambda_j \tfrac{\partial g_j}{\partial x_i} = 0,\ i = 1, 2, ..., n \\ &\tfrac{\partial L}{\partial \lambda_j} = 0, \Rightarrow g_j(x^*) = 0,\ j = 1, 2, ..., m \end{aligned} \tag{2.7}$$

  Eşitlik 2.7'deki denklemler Eşitlik 2.8 ve Eşitlik 2.9'daki gibi ifade edilebilir.

$$\nabla L(x^*, \lambda^*) = 0,\ veya\ \frac{\partial L(x^*, \lambda^*)}{\partial x_i} = 0,\ i = 1, ..., n \tag{2.8}$$

$$\frac{\partial L(x^*, \lambda^*)}{\partial \lambda_k} = 0\ \Rightarrow g_j(x^*) = 0,\ j = 1, ..., m \tag{2.9}$$

Eşitlik 2.8 ve Eşitlik 2.9'un gradyent koşulları Lagrange fonksiyonunun hem x hem de λ'ya göre durağan bir fonksiyon olduğunu göstermektedir. Böylelikle, Lagrange fonksiyonu değişimin az olduğu durağan noktaları belirlemek için x ve λ değişkenli sınırlamasız bir fonksiyon olarak ele alınır. Burada teorem koşullarını sağlamayan noktalar yerel minimum olamaz. Ancak şartları sağlayan bir noktanın da yerel minimum olduğu garanti edilemez. Aslında bu noktalar aday noktalardır ve yeter şartları sağlama durumuna göre karar verilir.

Eşitlik 2.7'de verilen denklemler çözülerek Eşitlik 2.10'de verilen $f(x)$'in çözüm vektörü ve Lagrange çarpanları vektörü elde edilir.

$$\vec{x}^* = \begin{Bmatrix} x_1^*, \\ x_2^*, \\ . \\ . \\ . \\ x_n^* \end{Bmatrix} \quad \vec{\lambda}^* = \begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_n \end{Bmatrix} \tag{2.10}$$

Geometrik olarak gerek şartın anlamına gelirsek, gerek şart sonucu elde edilen aday noktada, amaç fonksiyonunun gradyantı ile sınırlama fonksiyonunun gradyantı aynı doğru üzerindedir ve Lagrange çarpanı bu ikisinin arasındaki oranı belirler.

$f(x)$'in x^* noktasında yerel minimuma sahip olması için yeter şart kuadratikliktir (Q) ve her $x = x^*$ noktasında sınırlamaları karşılayan dx'in tüm değerleri için pozitif olmalıdır.

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j}(x^*, \lambda^*) dx_i dx_j$$

2.2.2. Eşitsizlik Sınırlamalı Çok-Değişkenli Optimizasyon

$$minimize\ f(\vec{x}) \tag{2.11}$$

$$g_j(\vec{x}) \leq 0,\ j = 1, 2, ..., m \tag{2.12}$$

Eşitsizlik sınırlamalı problemi için sınırlamaya negatif olmayan, değeri bilinmeyen serbest (slack) bir değişken y_j^2 eklenerek eşitlik sınırlamasına dönüştürülür. Bu problem daha önce anlatılan yöntemlerden Lagrange çarpanı metodu ile çözülebilir. Problemin son halinde sınırlama $G(x, y) = g_j(\vec{x}) + y_j^2 = 0,\ j = 1, 2, ..., m$ şeklindedir. $\vec{y} = \{y_1, y_2, ..., y_m\}^T$ serbest (slack) değişkenler vektörüdür.

$$L(x, y, \lambda) = f(x) + \sum_{j=1}^{m} \lambda_j G_j(x, y) \tag{2.13}$$

Lagrange fonksiyonunun aday noktaları aşağıdaki denklemler çözülerek bulunur (gerek şart).

$$\begin{aligned} &\frac{\partial L}{\partial x_i}(x, y, \lambda) = \frac{\partial f}{\partial x_i}(x) + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i}(x) = 0, i = 1, 2, ..., n \\ &\frac{\partial L}{\partial \lambda_j}(x, y, \lambda) = G_j(x, y) = g_j(x) + y_j^2 = 0, j = 1, 2, ..., m \\ &\frac{\partial L}{\partial y_j}(x, y, \lambda) = 2\lambda_j y_j = 0, j = 1, 2, ..., m \end{aligned} \tag{2.14}$$

Görüldüğü üzere denklemlerde toplam $n + 2m$ bilinmeyen bulunmaktadır. Bu denklemlerin çözümünde optimum çözüm vektörü $\vec{x}$, Lagrange çarpanı vektörü $\vec{\lambda}$ ve serbest (slack) değişken vektörü $\vec{y}$ elde edilir.

Kuhn-Tucker Gerek Şartları: Eşitlik sınırlamalı problemler için gerek şartlar genel itibariyle *Kuhn-Tucker şartları* olarak adlandırılır. Bu şartlar genellikle yerel minimumluğu garanti etmekte yeterli değildir. Ancak konveks programlama problemleri olarak adlandırılan problem grubunda Kuhn-Tucker şartları küresel minimum için gerek ve yeter şartlardır. Kuhn-Tucker şartlarının uygulanabilmesi için problemin sınırlamalarının aşağıdaki özelliklerden birini sağlaması gerekir. Bu özelliklere *sınırlama kalitesi* denir. Bunlar:

- Bütün eşitsizlik ve eşitlik sınırlama fonksiyonları lineer olmalıdır.

- Bütün eşitsizlik sınırlama fonksiyonları konveks, bütün eşitlik sınırlama fonksiyonları lineer olmalı ve kabul edilebilir bölge içinde en az bir kabul edilebilir $\vec{x}$ vektörü olmalıdır.

Aktif sınırlamalar bilinmezse, Kuhn-Tucker şartları aşağıdaki şekilde ifade edilir:

$$\begin{aligned} \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} &= 0, \quad i = 1, 2, ..., n \\ \lambda_j x_j &= 0, \quad j = 1, 2, ..., m \\ g_j &\leq 0, \quad j = 1, 2, ..., m \\ \lambda_j &\geq 0, \quad j = 1, 2, ..., m \end{aligned} \tag{2.15}$$

Eğer problem maksimizasyon problemi ise veya sınırlamalar $g_j \leq 0$ formunda ise λ_j Eşitlik 2.15'te pozitif olmayan bir değer olmalıdır. Diğer yandan, problem $g_j \leq 0$ sınırlamasına sahip minimizasyon problemiyse, λ_j negatif olmayan bir değer olmalıdır [3].

Kuhn-Tucker şartları iki amaçla kullanılır: 1) Aday minimum noktaların tespiti (yerel minimum). 2) Bulunan çözümün optimum nokta olup olmadığını belirleme.

Kuhn-Tucker gerek şartıyla ilgili önemli özellikler [2]:

- Kuhn-Tucker şartları sadece düzenli noktalara uygulanabilir.
- Kuhn-Tucker şartını sağlayan noktalar Kuhn-Tucker noktaları olarak adlandırılırlar. Kuhn-Tucker şartını sağlamayan noktalar eğer düzensiz noktalar değilse yerel minimum olamazlar.
- Kuhn-Tucker şartlarını sağlayan noktalar sınırlamalı yada sınırlamasız olabilir.
- Eşitlik sınırlamaları varsa ve eşitsizlik sınırlamalarının hiçbiri aktif değilse, şartları sağlayan noktalar sadece aday noktalardır. Yani yerel minimum, maksimum yada dönüm noktalarıdır.

2.3. Sınırlamaları Ele Alış Metotları

Sınırlamalı optimizasyon problemlerinin çözümü zor olduğu için analitik optimizasyon metotları kabul edilebilir çözümleri bulmada yetersiz kalmıştır. Çünkü geleneksel optimizasyon metotları amaç fonksiyonunun türevlenebilirliği ve sürekliliği üzerine güçlü varsayımlar yapar [11]. Modern optimizasyon yöntemleri ise sınırlama ele alış (constraint-handling) yöntemleri ile türev bilgisi gerektirmeden sınırlama ve amaç fonksiyonu uzayının yapısı ile ilgili kabuller yapılmasını gerektirmeden sınırlamalı problemleri çözebilmekte ve etkin sonuçlar üretebilmektedir. Sınırlama ele alış metotları Michalewicz ve Schoenauer [12] tarafından dört kategoriye ayrılmıştır.

1. Çözümleri kabul edilebilir bölgede tutan metotlar
2. Ceza terimine dayalı metotlar
3. Kabul edilebilir ve kabul edilebilir olmayan çözümler arasında ayrım yapan metotlar
4. Karma metotlar

Bu metotlardan tez çalışmasında kullanılanlara ayrıntılı olarak değinilecektir.

2.3.1. Ceza Fonksiyonları

Bu metodun temel düşüncesi, bir çözümde var olan sınırlama ihlal miktarına göre amaç fonksiyonuna bir değer ekleyerek yada amaç fonksiyonundan bir değer çıkararak sınırlamalı optimizasyon problemini sınırlamasıza çevirmektir.

Sınırlamalı optimizasyon problemlerini çözmek için yaygın bir yaklaşım sınırlama ihlallerini cezalandırmak için 2.16'da gösterildiği gibi amaç fonksiyonuna bir ceza terimi eklemektir. Böylece sınırlamalı olan problem sınırlamasız optimizasyon problemine dönüştürülmüş olur.

$$\psi(x) = f(x) + r_g\phi(g_j(x); \qquad j = 1, 2, ..., m) \tag{2.16}$$

Analitik optimizasyonda ceza fonksiyonlarının iç (interior) ve dış (exterior) olmak üzere iki çeşiti vardır. Dış metotlarda, kabul edilebilir olmayan çözümle başlanır ve oradan kabul edilebilir bölgeye doğru taşınır. İç metotlarda değeri sınırlamaların sınırından uzaklaştıkça küçülecek veya sınırlamaların sınırlarına yaklaştıkça sonsuza gidecek bir ceza terimi belirlenir. Eğer kabul edilebilir bir noktadan başlanırsa, bir sonraki nokta da her zaman kabul edilebilir bölge içinde olacaktır. Çünkü sınırlamaların sınırları optimizasyon süreci boyunca bariyer görevi görür.

En yaygın kullanılan sınırlama ele alış metotlarından biri dış ceza yaklaşımıdır. Bunun temel sebebi ise dış ceza metodunun başlangıçta kabul edilebilir çözüm gerektirmemesidir. İç ceza yaklaşımının gerektirmesi ise dezavantajıdır.

Dış ceza fonksiyonunun genel formülasyonu aşağıda Eşitlik 2.17'de verilmiştir.

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right] \tag{2.17}$$

Bu formülasyonda $\phi(\vec{x})$ optimize edilecek yeni amaç fonksiyonudur. G_i ve L_j sınırlamaların $g_i(\vec{x})$ ve $h_j(\vec{x})$ fonksiyonlarıdır. r_i ve c_j ceza katsayıları olarak adlandırılan pozitif sabitlerdir.

G_i ve L_j'nin en yaygın şekli:

$$\begin{aligned} G_i &= max[0, g_i(\vec{x})]^{\beta} \\ L_j &= |h_j(\vec{x})|^{\gamma} \end{aligned} \tag{2.18}$$

β ve γ değerleri 1 veya 2'dir.

Ceza fonksiyonları hem eşitlik hem de eşitsizlik sınırlamalarını ele alır. Eşitlik sınırlamalarındaki yaklaşım ise sınırlamayı eşitsizlik sınırlamasına çevirmektir.

$$|h_j(\vec{x})| - \varepsilon \leq 0 \tag{2.19}$$

Burada ε çok küçük bir tolerans değeridir.

Ceza fonksiyonu türleri aşağıda açıklanmıştır [13].

- **Ölüm (Death) Cezası:** Kabul edilebilir olmayan çözümler direkt reddedildiği bu metot en basit ve aynı zamanda hesaplama açısından da verimli bir sınırlama ele

alış metodudur. Sınırlamayı ihlal eden çözüm reddedilirek yeniden oluşturulduğundan kabul edilebilir olmayan çözüm için ileri derece hesaplamalara gerek kalmaz. Kabul edilebilir bölgenin çok büyük olduğu veya konveks olduğu durumlarda önerilmez.

- **Statik Ceza:** Bu yaklaşım rasgele (kabul edilebilir veya kabul edilebilir olmayan) çözümler ile başlar. Bir birey aşağıdaki Eşitlik 2.20 kullanılarak değerlendirilir.

$$f^{'}(x) = f(x) + \sum_{i=1}^{m} (R_{k,i} \times max\left[0, g_i(x)\right]^2) \tag{2.20}$$

Bu eşitlikte $R_{k,i}$ kullanılan ceza katsayısı, m toplam sınırlama sayısı, $f(x)$ cezalandırılmamış amaç fonksiyonu ve l ($k = 1, 2, ..., l$) kullanıcı tarafından tanımlanan ihlal seviyesi sayısıdır. Bu yaklaşımın ana fikri, deterministik kurallar aracılığıyla çözümlerin sınırlamalarının her birine farklı bir faktör tanımlayarak çözümleri dengelemektir. Ceza faktörleri mevcut jenererasyon sayısına bağlı olmadan tüm iterasyonlar boyunca sabit kalmaktadır. Sınırlamaların ihlal seviyesi tanımlanarak ceza faktörleri ile ilişkilendirilir. Dezavantajlarından biri iterasyonlar boyunca aynı ceza faktörünü tutmaktır. Ceza faktörleri genel olarak probleme bağlı olarak değişir. Yaklaşım basittir ancak bazı durumlarda kullanıcının ceza faktörünü yüksek değerde kullanması gerekebilir.

- **Dinamik Ceza:** Bu türde mevcut iterasyon sayısına karşılık gelen ceza faktörü hesaplanmaktadır. Ceza fonksiyonu zamanla yani her iterasyonda artacak şekilde tanımlıyken Joines ve Houck [14], Eşitlik 2.21'i kullanarak t iterasyonda değerlendirdiği çözümler için bir teknik önermiştir.

$$fitness(\vec{x}) = f(\vec{x}) + (C \times t)\alpha \times SVC(\beta, \vec{x}) \tag{2.21}$$

Bu eşitlikte C, α ve β kullanıcı tarafından tanımlanan sabit sayılardır.

$$\begin{aligned} SVC(\beta, \vec{x}) &= \textstyle\sum_{i=1}^{n} D_i^{\beta}(\vec{x}) + \sum_{j=1}^{p} D_j(\vec{x}) \\ D_i(\vec{x}) &= \begin{cases} 0, & g_i(\vec{x}) \leq 0 \\ \left|g_i(\vec{x})\right|, & diger \end{cases} \quad 1 \leq i \leq n \\ D_j(\vec{x}) &= \begin{cases} 0, & -\varepsilon \leq h_j(\vec{x}) \leq \varepsilon \\ \left|h_j(\vec{x})\right|, & diger \end{cases} \quad 1 \leq j \leq p \end{aligned} \tag{2.22}$$

SVC dinamik fonksiyonu iterasyon ilerledikçe cezayı artırır. Bazı çalışmalarda dinamik cezanın daha iyi olduğu öne sürülmüştür. Ancak dinamik olarak ceza

fonksiyonu oluşturmak statik olarak ceza faktörü oluşturmak kadar zordur [15]. Joines ve Houck [14]'un sunduğu yaklaşıma göre bulunan çözümün kalitesi α ve β değerlerindeki hassas değişime bağlıdır ve C'nin farklı değerleri için yaklaşımın hassaslığına ilişkin açık bir sonuç yoktur. Dinamik ceza fonksiyonlarının probleme bağlı olarak performansı değişebilir. Statik cezanın dezavantajları dinamik ceza fonksiyonlarında da bulunmaktadır. Eğer ceza faktörü iyi seçilmezse yani çok yüksek seçilirse algoritmanın optimum olmayan kabul edilebilir çözümlere yakınsamasına yada çok küçük seçilirse kabul edilebilir olmayan çözümlere yakınsamasına sebep olur.

- **Adaptif Ceza:** Araştırma esnasında geri bildirim alarak çarpan değerini değiştiren bir ceza fonksiyonu kullanır. Her çözümün uygunluğu Eşitlik 2.23 ile değerlendirilir.

$$fitness(x) = f(x) + \lambda(t)\left[\sum_{i=1}^{n} g_i^2(x) + \sum_{j=1}^{p} |h_j(x)|\right] \tag{2.23}$$

$\lambda(t)$ değeri her t iterasyonda Eşitlik 2.24 ile güncellenir.

$$\lambda(t+1) = \begin{cases} (1/\beta_1).\lambda(t), & durum-1 \\ \beta_2.\lambda(t), & durum-2 \\ \lambda(t), & diger\ durumlar \end{cases} \tag{2.24}$$

Son k jenerasyondaki en iyi çözüm kabul edilebilir ise *durum-1*, kabul edilebilir değilse *durum-2* uygulanır. (Döngüyü engellemek için $\beta_1 > 1$, $\beta_2 > 1$, $\beta_1 > \beta_2$ ve $\beta_1 \neq \beta_2$ olmalıdır.) Yani $(t+1)$. jenerasyon için $\lambda(t+1)$ ceza bileşeni, son k jenerasyondaki tüm en iyi çözümlerin hepsi kabul edilebilir ise azalır, hepsi kabul edilebilir değilse artar. Bazı çözümler kabul edilebilir bazıları değilse bu durumda ceza değişmez.

Bu yaklaşımdaki dezavantaj k değerinin belirlenmesi gibi parametrelerin ayarlanmasındaki zorluk olabilir. Bu yaklaşımın ilginç yanı ise tamamen kabul edilebilir yada tamamen kabul edilebilir olmayan çözümlerden kaçınmasıdır.

- **Isıl İşlem Cezası:** Yapay ısıl işlemin mantığına dayanan yöntemde yerel minimuma takılma sorunu engellemek için aktif sınırlamalar her iterasyonda dikkate alınarak ceza zamanla (yani sıcaklık zamanla arttıkça) artırılmaktadır. Böylelikle kabul edilebilir olmayan çözümler ağırlıkla son jenerasyonlarda cezalandırılmaktadır.

2.3.1.1. Stokastik Sıralama

Ceza fonksiyonu metodu bazı problemler için iyi sonuçlar vermektedir. Ancak r_g için optimal değere karar vermek de bir optimizasyon problemi olarak karşımıza çıkmaktadır. Çünkü r_g değeri eğer çok küçük seçilirse, kabul edilebilir olmayan çözümler yeterince cezalandırılmamış olabilir. Bundan dolayı, kabul edilebilir olmayan çözümler algoritma tarafından çözümlere dahil edilebilir. Eğer r_g değeri çok büyük olursa, kabul edilebilir çözümle muhtemele bulunur ancak çok düşük kalitede olabilir. Çok büyük r_g değeri daha ilk iterasyonlarda, kabul edilebilir olmayan bölgelerin araştırılmasını engeller. Bu durum kabul edilebilir bölgeleri araştırma uzayında parçalı halde bulunan problemler için elverişsizdir. Burada kabul edilebilir olmayan bölgelerin ne kadar araştırılacağı önemli bir husustur. Bu değer probleme bağlı olarak değişmektedir. Aynı problem için bile farklı iterasyonlarda farklı r_g değeri gerekebilir.

Optimal r_g'yi belirlemek zor olduğundan amaç ve ceza fonksiyonunu dengelemek için Runarsson ve Yao [16], amaç ve ceza fonksiyonların baskınlığına göre ceza fonksiyonları için yeni bir yaklaşım önermişlerdir.

Genel bir nonlineer programlama probleminin formülasyonu Eşitlik 2.1'de verilmiştir ve bu eşitlikte parametre sınırları ile sınırlanan n boyutlu uzayda, $x \in S \cap F, S \subseteq R^n$ araştırma uzayını tanımlar. F kabul edilebilir bölge Eşitlik 2.25 ile tanımlanmıştır ve burada $g_j(x), \quad j \in 1, 2, ..., m$ sınırlamaları ifade eder.

$$F = \{x \in R^n \mid g_j(x) \leq 0 \qquad \forall j \in \{1, 2, ..., m\}\} \tag{2.25}$$

Sıralama *kabarcık sıralama* benzeri bir prosedür ile yapılır. Bu prosedür sıralanmış bir dizideki baskınlığı dengelemenin en uygun yoludur. λ çözüm en az λ karşılaştırma yapılarak sıralanır ve sıralamada herhangi bir değişiklik olmadığında durur. Popülasyondaki çözümleri sıralamak için kullanılan stokastik kabarcık sıralama prosedürünün sözde kodu Algoritma 2.1'de verilmiştir. Bu yaklaşımda araştırma uzayının kabul edilebilir olmayan bölgelerindeki sıralamada karşılaştırmalar için sadece amaç fonksiyonunun kullanılma olasılığı P_f işleme dahil edilmiştir. Yani verilen iki komşu çözüm için her ikisi de kabul edilebilir bölgede ise amaç fonksiyonuna

göre onları karşılaştırma (hangisinin daha uygun olduğunu belirlemek için) olasılığı 1'dir. Aksi taktirde P_f'dir. Stokastik sıralama tekniğinde sıralama tabanlı bir seçim kullanılır ve kendinden ayarlanan P_f için ekstra bir hesaplama maliyeti yoktur. Daha önemlisi stokastik sıralama fikri optimizasyonda direkt ve açık bir şekilde amaç ve ceza fonksiyonlarını dengeleme ihtiyacından gelir.

Algoritma 2.1: Kabarcık sıralama prosedürü

```
1: I_j = j ∀j ∈ {1, 2, ..., λ}
2: for i = 1 to N do
3:     for j = 1 to λ − 1 do
4:         sample u ∈ U(0, 1)
5:         if (φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f) then
6:             if f(I_j) > f(I_{j+1}) then
7:                 swap(I_j, I_{j+1})
8:             end if
9:         else
10:            if φ(I_j) > φ(I_{j+1}) then
11:                swap(I_j, I_{j+1})
12:            end if
13:        end if
14:    end for
15: end for
16: if noswap then
17:    break
18: end if
```

Bu prosedürde $U(0,1)$ uniform rasgele sayı üretici, N tüm popülasyon boyunca karşılaştırmaların sayısıdır. $P_f = 0$ ise sıralama aşırı cezalandırılmış, $P_f = 1$ ise sıralama eksik cezalandırılmış demektir.

2.3.2. Deb Kurallarına Dayalı Seçim Stratejisi

Kabul edilebilir ve kabul edilebilir olmayan çözümler arasında ayrım yapan metotlardan biridir. Seleksiyon aşamasında aşağıdaki üç adıma göre kıyaslama yaparak bir seçim yapar ve çözümleri kabul edilebilir bölgeye yaklaştırır. [17].

- Her kabul edilebilir çözüm $(ihlal_i \leq 0)$ kabul edilebilir olmayan çözüme $(ihlal_j > 0)$ tercih edilir.(Çözüm i baskın)
- İki kabul edilebilir çözüm $(ihlal_i \leq 0, ihlal_j \leq 0)$ arasında amaç fonksiyon değeri daha iyi olan tercih edilir. ($f_i < f_j$, çözüm i baskın)

- İki kabul edilebilir olmayan çözüm $(ihlal_i > 0, ihlal_j > 0)$ arasında sınırlama ihlal değeri küçük olan tercih edilir. ($ihlal_i < ihlal_j$, çözüm i baskın)

2.4. Sınırlamaları Ele Alış Metotlarına Dayalı Modern Sezgisel Algoritmalar ile Yapılan Çalışmalar

Evrimsel Algoritmalar (EA), Evrimsel Stratejiler (ES), Diferansiyel Gelişim (DE), Parçacık Sürüsü Optimizasyonu (PSO), ve Yapay Arı Koloni (ABC) algoritması sınırlamalı optimizasyon problemlerini (SOP) çözmek için uyarlanarak bir çok çalışmada yer almıştır.

1. Evrimsel Algoritmalar (EA): Wu vd. EA kullanıldığında sınırlamalı optimizayon problemlerindeki değişkenlerin yanı sıra eşitlik sınırlamalarını azaltmak için eşitlik sınırlama ve değişken azaltma stratejisi (ECVRS) adını verdikleri bir strateji önermiştir. ECVRS özünde şunu ifade eder: SOP'da bazı değişkenler, yine başka değişkenlere bağlıdır ve yine hesaplama gerektirir. Bu durum EA'nın veriminin azalmasına ve araştırma uzayının daralmasına neden olur. ECVRS işte bu değişkenler arasındaki ilişkinin azaltılmasını ve böylece elde edilen uygun çözümlerin geliştirilmesini sağlar [18]. Abo-Bakr ve Mujeed, SOP için PSO ve GA tabanlı hibrit bir evrimsel algoritma önermiştir [19].

2. Evrimsel Stratejiler (ES): Runarsson ve Yao, amaç ve ceza fonksiyonlarını dengelemek için stochastic ranking (SR) kullanan bir yaklaşım sunmuştur [16]. Mezura-Montes ve Coello, SOP çözmek için bir çeşitlilik mekanizması ile birleştirilen basit ES'nin kullanımını sunmuştur. Önerilen mekanizma Niched-Pareto Genetik Algoritması (NGPA) olarak adlandırılan bir yaklaşımdan alınan çok amaçlı optimizasyon kavramına dayanır. Bu mekanizmanın başlıca avantajı her evrimsel stratejiden beklenenin dışında ekstra parametre tanımlanmasına ihtiyaç duymamasıdır [20].

3. Diferansiyel Gelişim Algoritması (DE): Liu vd. nümerik sınırlamalı optimizasyon problemlerini çözmek için DE ve PSO algoritmalarını birleştirerek adına PSO-DE adını verdiği hibrit algoritmayı geliştirdi [21]. Wang ve Cai geliştirilmiş adaptif karşılaştırmalı model ve $(\mu + \lambda)$-diferansiyel gelişim önermiştir [22]. Mallipeddi and Suganthan sınırlamalı optimazyon problemlerini çözmek için sınırlama ele alış tekniği topluluğu

(Ensemble of Constraint Handling Techniques (ECHT)) önermiştir. Her sınırlama metodu kendi popülasyonuna sahiptir. Burada ECHT'nin ayırt edici farkı her sınırlamaları ele alış tekniğiyle ilişkili her bir popülasyonun her fonksiyon çağrısını kullanmasıdır [23]. Mezura-Montes vd. sınırlamalı optimizasyon problemlerini çözmek için uyarlanan DE algoritmasının farklı versiyonlarının performanslarını deneysel sonuçlarla sunmuştur [24]. Sardar vd. $DE/rand/1/bin$ şeması ve gradyent tabanlı onarım fikrini birleştirerek DE algoritmasına uyarlamışlardır. DE algoritması tarafından üretilen aday birey çözümleri uygun değilse, bunları uygun çözümlere çevirmek için granyent tabanlı onarım uygulanmıştır [25]. Mohamed ve Sabry, Constrained Optimization based on Modified Differential Evolution algorithm (COMDE) yani değiştirilmiş DE algoritmasına dayanan sınırlamalı optimizasyonu önermiştir. Bu yeni algoritma belirli jenerasyondaki en iyi ve en kötü birey arasındaki ağırlıklandırılmış fark vektörüne dayanan yeni yönlendirilmiş mutasyon oranını kullanır [26]. Jia vd. $(\mu + \lambda)$- sınırlamalı diferansiyel gelişimin (CDE) temel sorunlarının üstesinden gelmek için ICDE ismini verdikleri $(\mu+\lambda)$-CDE'nin geliştirilmiş versiyonunu sunmuştur. ICDE temel olarak geliştirilmiş $(\mu+\lambda)$-diferansiyel gelişim (IDE) ve yeni depolama-tabanlı adaptif karşılaştırmalı model (ArATM)'den oluşur [27]. Gao vd. sınırlamalı optimizasyon problemleri için birlikte gelişen çift popülasyonlu diferansiyel gelişimi (DPDE) önermiştir. Sınırlamalı optimizasyon problemleri çift amaçlı optimizasyon problemi olarak ele alınır. Burada ilk hedef amaç fonksiyonunu optimize etmektir. İkinci hedef ise sınırlama ihlallerinin seviyesini açıklamaktır. Evrimsel süreç boyunca her jenerasyonda tüm popülasyon iki amacı sağlamak için uygun çözümlere dayanarak ikiye bölünür. Her bir alt popülasyon sadece ilgili amacını optimize etmeye odaklanır [28].

4. Parçacık Sürü Optimizasyonu (PSO): Zheng vd. lineer olmayan sınırlamalı optimizasyon problemleri için geliştirilmiş PSO (IPSO) önermiştir. IPSO'ya yeni bir mutasyon operatörü ve yeni araştırma uzayı keşfi için parçacıkları yönlendirmek amacıyla çoklu alt popülasyondan bazı komşu bireyleri birleştiren yeni bir yöntem uyarlanmıştır. Ayrıca sınırlamaları ele almak için basit ve kolay ceza fonksiyonlarını kullanır. Böylece algoritmanın sınırlamalı optimizasyon problemlerini çözerken güçlü küresel keşif yeteneği ve verime sahip olduğu üzerinde çalışılmıştır [29]. Sun vd. COP çözmek için yeni vektör parçacık sürü optimizasyonunu (NVPSO) önermiştir. NVPSO'da

bir parçacığın tüm boyutlarda alt ve üst sınırlar içinde olduğunun sağlanması için bir büzülme katsayısı ve parçacığın uygun bölgede olum olmadığına karar vermek için yeni bir fonksiyon sunulmuştur. Bir-boyutlu arama optimizasyon yöntemleri uygun bölgeden uzaklaşan parçacıkların uygun bölgede olmasını garanti eden yeni bir pozisyon üretmek için NVPSO algoritmasında kullanılmıştır. Tüm işlemler vektör modunda ele alınmıştır [30]. Miano vd. uygun bir bölgeye yakınsaması zor olan sınırlamalı optimizasyon problemleri için PSO aracılığıyla geliştirilen iki-aşamalı deneme yöntemini önermiştir. İlk aşama parçacığın mümkün olduğu kadar az olasılıkla uygun bölgeden uzaklaşmasını garanti eder. İkinci aşama ise Rosenbrock yöntemiyle yerel minimumlardan kurtulmayı amaçlar [31]. Sun vd. sınırlamalı optimizasyon problemlerinin çözümünde uygulanabilirlik kurallarına dayanan geliştirilmiş PSO (IPSO)'yu önermiştir. Bu algoritmada sürünün ortalama hızı ve parçacıkların komşuluğundaki geçmişteki en iyi pozisyonu iki türbülans faktörü olarak görülür. Türbülans faktörlerinin parçacıkların uçuş yönünü etkilediği düşünülür ve ilk başlarda yakınsamayı engellemek için dahil edilmez [32]. Wei ve Zhang, sınırlamalı çok amaçlı optimizasyon problemleri için ince ayarlanmış yerel arama operatörlü PSO'nun küresel arama yeteneği ile birleştirdiği yeni bir memetik algoritma sunmuştur. İlk olarak yeni parçacık güncelleme stratejisi erken yakınsama probleminin üstesinden gelmek için belirsiz kişisel en iyi kavramını ele almıştır. Yerel arama operatörüne dayanan çekim, parçacıklar için iyi bir yerel arama yönü bulmak için kullanılmıştır [33]. Li vd. SOP çözümü için dinamik komşuluklu PSO'yu önermiştir. Sınırlamaları ele alırken baskın sınırlama kavramını kullanmıştır. Dinamik komşuluğun amacı arama performansını artırmak için farklı gruplardan parçacıklar arasındaki iletişimi geliştirmektir [34]. Liu ve Hui, sınırlamalı optimizasyonda eşitlik ve eşitsizlik sınırlamalarının üstesinden gelmek için yeni bir yöntem geliştirmiştir. PSO algoritması sınırlamalı optimizasyonda sınırlamasız optimizasyonda olduğu kadar iyi değildir. Çünkü PSO uygun bölgeyi bulmakta fazla başarılı olamamıştır. PSO'nun bu zayıf yönünden yola çıkarak uygun bölgeyi bulmak için nümerik gradyent (NG) olarak adlandırılan yeni bir yöntem geliştirilmiştir. NG'nin sağladığı bilgi ile PSO problemin optimum pozisyonunu bulmuştur. Oluşturulan yeni yöntem nümerik granyent PSO (NGPSO) olarak adlandırılmıştır [35]. Elsayed vd. SOP çözerken PSO'nun yerel optimuma takılmaması için iki yeni mekanizma kullanmıştır. Bunlardan ilki yoğunlaştırma ve çeşitlendirme arasında daha iyi denge sağlanması diğeri ise yerel

çözümlerden uzaklaşmanın sağlanmasıdır [36]. Wei ve Jia, sınırlamalı dinamik çok amaçlı optimizasyon problemlerini çözmek için hem küresel optimum çözümü bulmayı sağlayan hem de dinamik çevre üzerinde değişen optimum noktayı izlemeyi gerektiren bir algoritma gerektiğini savunmuştur. Bu amaçla, böyle problemlerin çözümü için yeni PSO'yu geliştirmiştir. Bu geliştirilen algoritma evrimsel süreci hızlandırmak için yeni nokta seçim stratejisi ve muhtemel çözümlerin olduğu düşünülen bir alt bölgede optimum çözümü bulmak için bir yerel arama operatörü kullanır [37]. Wu, sınırlamalı küresel optimizasyon problemi için kuantum davranışlı PSO (QPSO) önermiştir. Standart PSO'nun yeteneklerini geliştirmek için popülasyondaki parçacıkların çeşitliliğini artırmak için bir Cauchy mutasyon operatörü, daralma genleşme katsayısını güncellemek için evrimsel jenerasyonlara dayanan bir operatör ve güçlü parçacıkları sürdürmek için elitist bir strateji geliştirmiştir [38].

5. Yapay Arı Koloni Algoritması (ABC): Karaboga ve Akay, SOP çözümü için ABC algoritmasının genişletilmiş versiyonunu önermiştir. Bu genişletilmiş algoritmada ABC'nin temel versiyonuna ek olarak iki parametre eklenmiştir. Bunlardan ilki yakınsama hızını artırmak için optimizasyon parametrelerinden olası modifikasyonları kontrol eden modifikasyon oranı (MR), diğeri ise kaşif arı fazında kullanılan ve yapay kaşif üretimi için önceden belirlenmiş çevrim süresini belirten kaşif üretim periyodu (SPP)'dur. Genişletilmiş ABC sınırlamaları ele alırken seleksiyon aşamasında Deb'in kurallarını kullanır [39, 40]. Bacanin ve Tuba, SOP'un çözümüyle ABC'nin performansının artmasını sağlamak amacıyla ABC algoritmasının modifikasyonunu sunmuştur. Yapılan modifikasyonlar ise genetik algoritma (GA) operatörlerini baz alır ve yeni aday çözümlerin oluşturulması sırasında uygulanır [10]. Mezura-Montes ve Cetina-Domínguez, SOP için ABC algoritmasının farklı versiyonunu (M-ABC) geliştirmiştir. Orijinal ABC algoritmasına dört mekanizma eklenmiştir: turnuva seçimi, eşitlik sınırlamaları için dinamik tolerans, zeki uçuş operatörü ve sınır değerlerini ele alış [41]. Brajevic ve Tuba, SOP çözümü için yükseltilmiş yapay arı koloni (UABC) algoritmasını tanıtmıştır. Bu algoritma modifikasyon oranı parametresinin özelliklerinde düzenleme yaparak geliştirilmiş ve ABC algoritmasının değiştirilmiş kaşif arı fazını kullanmıştır [42]. Aguilar-Justo vd. sınırlamalı numerik optimizasyon problemlerine performansı artıran Hooke-Jeeves yöntemi ile değiştirilmiş ABC (MABC)

algoritmasını birleştiren memetik bir yaklaşım önermiştir. Görevli arı tarafından kullanılan operatör daha çeşitli çözümler elde etmek için değiştirilmiştir. Sınırlamaları ele almak amacıyla orijinal MABC'de kullanılan uygulanabilirlik kuralları seti ϵ-sınırlama yöntemiyle yer değiştirmiştir. Ayrıca yerel arama sıklığı geçerli popülasyondaki çözümlerin çeşitliliğine dayanan bir ölçüme bağlıdır [43]. Li ve Yin, uygulabilir kural metotu ve çok-amaçlı optimizasyon metotuna dayanan sınırlamalı optimizasyon problemi için self-adaptif sınırlamalı ABC algoritmasını tanıtmıştır. Bu algoritmaya göre görevli (işçi) arı kolonisi uygulanabilir kurallara dayanan her popülasyon için küresel arama motoru gibi davranır. Gözcü arı kolonisi ise çok amaçlı optimizasyona dayanan yeni araştırma uzayı keşfedebilir. Önerilen algoritmanın yakınsama oranını artırmak amacıyla, algoritmanın birçok parametresinin değiştirebilmesini sağlamak için self-adaptif bir modifikasyon oranı önerilmiştir [44]. Brajevic, çalışmasında SOP için ABC algoritmasının performansını artırmak amacıyla crossover-tabanlı yapay arı koloni (CB-ABC) algoritmasını önermiştir. ABC algoritmasının farklı alanlardaki başarısına rağmen küresel optimuma yaklaşım hızını artırmak için DE ve GA'da olduğu gibi crossover operatörünün kullanılması gerektiği düşünülmüştür. Bundan dolayı çözümler arasındaki iyi bilginin dağıtımının geliştirmek için keşif fazında rasgele arama yerine uniform crossover operatörü kullanılmıştır. Ayrıca mutasyon işleminde rasgele seçilen komşu çözümlerden dolayı çözümlerin keşfinde eksiklik olabileceğinden bu sorunu aşmak için CB-ABC sırasıyla görevli ve gözcü fazlarında iki değiştirilmiş ABC arama operatörü kullanır. Ek olarak önerilen algoritma limit sınırlama metotlarını geliştirmek ve eşitlik sınırlamalarını ele almak için dinamik toleransla ilgili iki değişiklik daha içerir [45]. Akay ve Karaboga, bu çalışmasında SOP için ABC algoritmasının farklı versiyonlarını karşılaştırmıştır. Bu versiyonlar komşuluk topolojisi ve farklı komşu üretim mekanizmalarını kullanır [46].

3. BÖLÜM

YÖNTEM

3.1. Yöntem

Bu tez çalışmasında Yapay Arı Kolonisi algoritmasında yapılacak güncellemeler ile sınırlamalı optimizasyon problemleri üzerindeki performans analiz edileceği için alt bölümlerde Yapay Arı Kolonisi algoritması anlatılacaktır.

3.2. Arıların Doğal Yaşamı ve Yiyecek Arama Davranışları

Doğadaki bal arıları dinamik olarak görev paylaşımı yapan ve çevredeki değişimlere karşı toplu bir şekilde yanıtlar vererek kendi kendine adapte olabilen ilginç sürülerden birisidir. Bal arıları fotografik hafıza, navigasyon sistemi, içgüdü yeteneği ve yeni yuva seçme işleminde toplu karar verebilme özelliklerine sahiptirler. Ayrıca bal ve polenlerin (yiyecekler) yuvaya getirilmesi, saklanması, dağıtılması, iletişim ve yiyecek arama gibi işleri de yürütürler [47]. Yiyecek arama hayatlarının devamı için en kritik görevlerden birisidir. Yiyecek arama davranışında işçi arılar 3 farklı rolde bulunabilirler: görevli, gözcü ve kaşif.

Yiyecek arama esnasında arılar arasında bilgi paylaşımı kollektif bilginin oluşumunda en önemli husustur. Yiyecek kaynağının kalitesi ve yeri ile ilgili bilgiler dans alanında görevli arılar tarafından diğer arılarla paylaşılır. Bir kovandaki en önemli alanlardan biri dans alanıdır. Ziyaret ettikleri kaynağa daha fazla arı gönderebilmek için kovandaki çeşitli alanlarda bu dansı gerçekleştirir ve kaynağına dönerler.

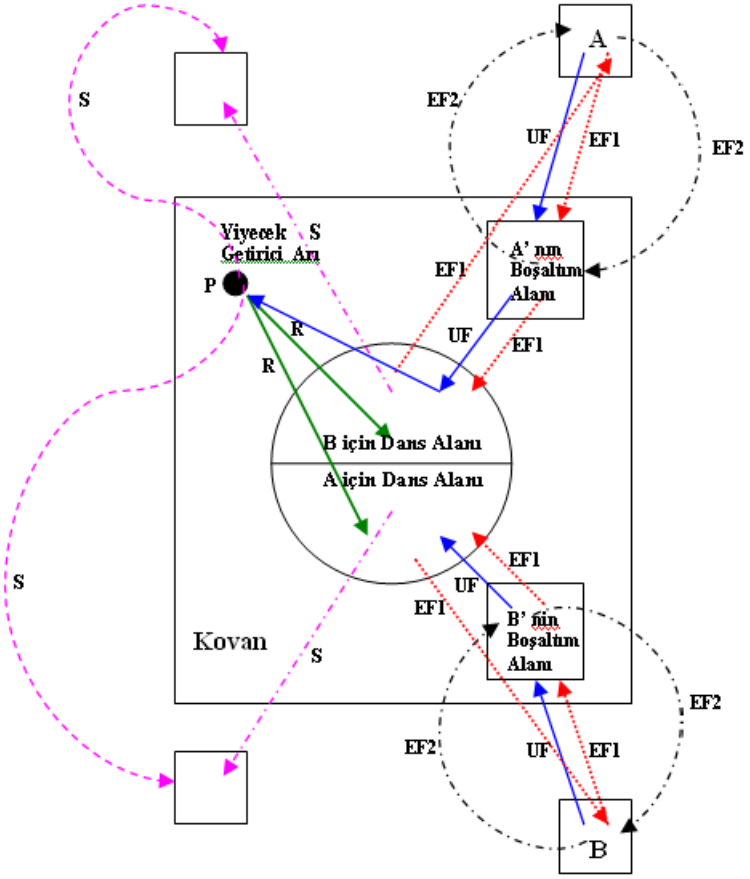
Arılar uzak bir noktadaki kaynağa diğer arıları yönlendirirken yön bilgisini vermeleri gerekir. Yön bilgisi alındıktan sonra hedefe ulaşmak için güneşten yararlanılır. Arılar göz

yapıları ile güneş ile kendi yörüngeleri arasındaki açıyı hesaplayabilirler. Ayrıca uzak noktaya gidip gelme sırasındaki enerji tüketimlerini de ayarlarlar.

Tüm zengin kaynaklarla ilgili bilgiler dans alanında dansları izleyen gözcü arılara iletildikten sonra gözcü arılar hangi kaynağı tercih edeceğine karar verir. Yiyecek arayıcıların davranışları Şekil 3.1'de incelenmiştir [1].

Keşfedilen iki kaynak A ve B olarak adlandırılır. Öncelikle görev atanmamış bir işçi arı kovan etrafındaki kaynakların yerini bilmeden yiyecek arayıcı olarak araştırmaya başlar. Bu arı için iki seçenek bulunur: i) Bu arı kaşif arı olarak yiyecek aramaya devam edebilir (Şekil 3.1'de S ile gösterilmektedir). ii) Bu arı gözcü arı olarak dansları izleyerek aldığı bilgi doğrultusunda kaynağa gidebilir (Şekil 3.1'de R ile gösterilmektedir).

Kaynak bulunduktan sonra kaynağın konumu arı tarafından hafızaya alınır ve nektar toplanır. Görev atanmamış arı böylelikle görevli arıya dönüşür. İşçi arı nektar toplama işlemi bittikten sonra kovana döner ve topladığı nektarları yiyecek depolarına aktarır. Nektarı topladıktan sonra arı için üç seçenek oluşur: i) Gittiği kaynağı bırakarak bağımsız izleyici olabilir (Şekil 3.1'de UF ile gösterilmektedir). ii) Kovandaki diğer arıları da aynı kaynağa yönlendirmek için dans edebilir (Şekil 3.1'de EF1 ile gösterilmektedir). iii) Hiçbir yönlendirme yapmadan kaynağına gidebilir (Şekil 3.1'de EF2 ile gösterilmektedir).



Şekil 3.1. Yiyecek Arama Çevrimi

3.3. Yapay Arı Koloni Algoritması

Yapay Arı Koloni (ABC) algoritması 2005 yılında Karaboğa [48] tarafından önerilen ve arıların yiyecek arama davranışını simüle eden bir optimizasyon algoritmasıdır. Arılar kendilerini organize edebilme ve kendi aralarında iş bölümü yapabilme yeteneğiyle kovanda depoladıkları yiyecek miktarını en yüksek seviyeye çıkarmaya çalışırlar. Kendi kendine organize olma ve iş bölümü sürü zekasının temel bileşenleridir. Tereshko [49]'ya göre arılarda kollektif zekanın ortaya çıkmasını sağlayan minimal yiyecek arama modeli üç temel bileşenden oluşur: yiyecek kaynakları, görevli işçi arılar ve görev atanmamış işçi arılar. Bu modelde yiyecek kaynağına yönelme ve kaynağı terketme olarak iki davranış vardır.

1. Yiyecek Kaynakları: Arıların yiyecek bulmak için gittiği kaynaklar yiyecek kaynaklarıdır. Bu kaynağın değeri; yuvaya yakınlığı, nektar yoğunluğu veya nektarın

çıkarılma kolaylığı gibi birçok etkene bağlıdır. Ancak kolaylık açısından tek kriter olarak kaynağın zengin olması alınabilir.

2. Görevli İşçi Arılar: Önceden keşfettiği yiyecek kaynağından nektarın kovana getirilmesinden sorumludur. Ayrıca ziyaret ettikleri yiyecek kaynağının kalitesi hakkında kovanda bekleyen diğer arılara da bilgi verirler.

3. Görev Atanmamış İşçi Arılar: Bu arıların nektarını çıkardığı kaynak henüz yoktur ve nektar toplayabileceği kaynak bulma eğilimindedir. Görev atanmamış iki arı çeşiti vardır: Kaşif arı ve gözcü arı. Kaşif arılar yeni bir yiyecek kaynağı bulmak için araştırma yapar. Bunu yaparken iç güdülerini veya dış etkileri rasgele kullanır. Gözcü arılar ise kovanda beklerler ve görevli arılardan gelen bilgiye göre ziyaret etmek için bir yiyeyek kaynağı seçerler.

Karaboğa [48] tarafından önerilen ABC algoritmasında yiyecek kaynaklarının yerleri optimizasyon problemine ait olası çözümlere ve yiyecek kaynağındaki nektar miktarı ise çözümlerin kalitesine (uygunluk değerine) karşılık gelmektedir. İşçi arıların veya gözcü arıların sayısı popülasyondaki yiyecek kaynağının sayısına eşittir. ABC algoritması, uzaydaki çözümlerden en fazla nektara sahip kaynağın yerini bulmaya çalışarak problemin maksimumunu veya minimumunu veren noktayı (çözümü) bulmaya çalışır.

ABC algoritmasının adımları Algoritma 3.1'de verilmektedir [1].

Algoritma 3.1: ABC algoritmasının temel adımları

```
1: Başlangıç yiyecek kaynağı bölgelerinin üretilmesi
2: repeat
3:     Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi
4:     Yiyecek Kaynaklarının Seçilebilme Olasılıklarının Hesaplanması
5:     Gözcü Arıların Yiyecek Kaynağı Bölgesi Seçmeleri
6:     Kaynağı bırakma kriteri: limit ve kaşif arı üretimi
7: until cevrimsayisi = maksimumcevrimsayisi
```

1. Başlangıç Yiyecek Kaynağı Bölgelerinin Üretilmesi: Algoritma kovan çevresini araştırma uzayı varsayarak rasgele yiyecek kaynağı yerleri oluşturur. Bu yerleri oluştururken her bir parametrenin alt ve üst sınırları dikkate alınarak rasgele değerler üretilir. Yiyecek kaynağı bölgelerinin üretilmesi Eşitlik 3.1 ile gerçekleştirilir:

$$x_{ij} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min}) \tag{3.1}$$

Bu eşitlikte SN yiyecek kaynağı sayısı ve D optimize edilecek parametre sayısı $i = 1, ..., SN$, $j = 1, ..., D$ olmak üzere x_j^{min}, j. parametrenin alt sınırı, x_j^{max} ise j. parametrenin üst sınırıdır. Aynı zamanda başlangıç aşamasında her kaynağın geliştirilememe sayısını ifade eden $sayac_i$ (i. kaynağın geliştirilememe sayısı) sayaçları da sıfırlanmaktadır. Başlangıç yiyecek kaynağı bölgeleri üretildikten sonra bu kaynaklar görevli arı, gözcü arı ve kaşif arı adımları ile devam ederek daha iyi bir kaynak bulunmaya çalışılır. ABC algoritmasında durdurma kriteri olarak kabul edilebilir hata değeri (ϵ), maksimum çevrim sayısı (MCN) veya standart durdurma kriteri uygulanabilir. Standart durdurma kriterleri evrimsel algoritmaların stokastik yapısından dolayı belirlenen maksimum CPU zamanının dolması, populasyon çeşitliliğinin belli bir eşik değeri altında kalması, toplam uygunluk hesaplamalarının belirlenen sayıya ulaşmış olması veya belirli zaman periyodu içinde uygunluk değeri gelişiminin bir eşik değeri altında kalması olabilir.

2. Görevli Arıların Yiyecek Kaynağı Bölgelerine Gönderilmesi: ABC algoritmasında her kaynağın sadece bir görevli arısı olabilir. Görevli arılar yeni bir yiyecek kaynağını, bulunduğu kaynağın komşuluğunda Eşitlik 3.2 ile belirler.

$$\upsilon_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

$\vec{x}_i$ ile gösterilen her bir kaynak için, bu kaynağın yani çözümün tek bir parametresi (rasgele seçilen parametresi, j) değiştirilerek $\vec{x}_i$ komşuluğunda $\vec{v}_i$ kaynağı bulunur. Eşitlik 3.2'de j, $[1, D]$ aralığında rasgele üretilen tamsayıdır. Rasgele seçilen j parametresi değiştirilirken, yine rasgele seçilen $\vec{x}_k$ komşu çözümünün ($k \in \{1, 2, ..., SN\}$) j. parametresi ile mevcut kaynağın j. parametresinin farkları alınıp, $[-1, 1]$ arasında değer alan ϕ_{ij} sayısı ile ağırlıklandırıldıktan sonra mevcut kaynağın j. parametresine eklenmektedir.

Eşitlik sonucu yeni üretilen $\vec{v}_{ij}$ daha önceden belli olan parametre sınırlarını aşarsa Eşitlik 3.3 ile j. parametreye ait olan alt veya üst sınır değerlerine çekilir.

$$\upsilon_{ij} = \begin{cases} x_j^{min} & \upsilon_{ij} < x_j^{min} \\ x_j^{max} & \upsilon_{ij} > x_j^{max} \end{cases} \tag{3.3}$$

Sınırlar dahilinde üretilen v_i parametre vektörü yeni bir kaynağı temsil etmekte ve bunun kalitesi Eşitlik 3.4 ile hesaplanarak bir uygunluk değeri atanmaktadır.

$$fitness_i = \begin{cases} 1/(1+f_i) & f_i \geq 0 \\ 1 + abs(f_i) & f_i < 0 \end{cases} \tag{3.4}$$

Burada f_i, $\vec{v}_i$ kaynağının yani çözümünün maliyet değeridir. $\vec{x}_i$ ile $\vec{v}_i$ arasında nektar miktarlarına yani uygunluk değerlerine göre bir aç gözlü (greedy) seçim işlemi uygulanır. Yeni bulunan $\vec{v}_i$ çözümü daha iyi ise görevli arı hafızasından eski kaynağın yerini silerek $\vec{v}_i$ kaynağının yerini hafızaya alır. Aksi taktirde görevli arı $\vec{x}_i$ kaynağına gitmeye devam eder ve $\vec{x}_i$ çözümü geliştiremediği için $\vec{x}_i$ kaynağı ile ilgili geliştirememe sayacı ($sayac_i$) bir artar, geliştirdiği durumda ise sayaç sıfırlanır.

3. Yiyecek Kaynaklarının Seçilebilme Olasılıklarının Hesaplanması: Görevli arıların hepsi bir çevrimde araştırmalarını bitirdikten sonra kovana dönerler ve buldukları kaynakların nektar miktarlarıyla ilgili bilgileri gözcü arılara iletirler. Gözcü arılar yiyecek kaynaklarının nektar miktarıyla orantılı bir olasılık kullanarak paylaşılan bilgiye göre bir kaynak seçer. Nektar miktarlarına denk gelen uygunluk değerleri hesaplanarak olasılıksal bir seçim yapılır. Bu seçim sıralamaya dayalı, stokastik, rulet tekerleği, turnuva yöntemi yada başka bir seçim yöntemlerinden istenilen ile yapılabilir. Temel ABC algoritması rulet tekerleği kullanarak seçim işlemini yapar. Tekerlekteki her bir dilimin açısı uygunluk değeriyle orantılıdır. Bir kaynağın uygunluk değerinin, tüm kaynakların uygunluk değeri toplamına oranı Eşitlik 3.5 ile bulunur ve bu oran nispi seçilme olasılığını verir.

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{3.5}$$

Burada $fitness_i$, $i.$ kaynağın kalitesinin, SN yiyecek kaynağı sayısını göstermektedir. Hesaplanan bu olasılık değerinden bir kaynağın uygunluk miktarı arttıkça yani nektar miktarı arttıkça, bu kaynağı seçecek gözcü arı sayısı da dolayısıyla artacaktır.

4. Gözcü Arıların Yiyecek Kaynağı Bölgesi Seçmeleri: Olasılık değerleri hesaplandıktan sonra her bir kaynak için $[0, 1]$ aralığında rasgele sayı üretilir. Rulet

tekerleğine göre seçim işleminde bu üretilen sayı p_i değerinden küçükse gözcü arılar tıpkı görevli arıların yaptığı gibi Eşitlik 3.2'yi kullanarak bu kaynak bölgesinde yeni bir çözüm üretir. Yeni çözüm değerlendirilir ve kalitesi hesaplanır. Yeni çözüm ve eski çözüm uygunluklarına göre karşılaştırılır. Aç gözlü seleksiyon işlemi kullanılarak iyi olan çözüm seçilir. Yeni çözüm seçilirse çözüm geliştirememe sayacı ($sayac_i$) sıfırlanır. Eski çözümün daha iyi olduğuna karar verilirse mevcut çözüm korunur ve gelistirememe sayacı ($sayac_i$) bir artırılır. Bu aşama tüm gözcü arılar yiyecek kaynağına yönlendirilene kadar sürer.

5. Kaynağı Bırakma Kriteri: Limit ve Kaşif Arı Üretimi: Tüm görevli ve gözcü arı aşamaları bir çevrimi bitirdikten sonra çözüm geliştirememe sayaçlarına ($sayac_i$) bakılır. Bir kaynağın nektarının tükenip tükenmediği bu sayaçlardan anlaşılır. Bir kaynak için çözüm geliştirememe sayacı belirlenen bir *limit* değerini aştıysa, bu kaynak tükenmiş demektir ve bu kaynağın görevli arısı kendine başka bir kaynak bulmalıdır. Yani görevli arının o kaynakla ilgili görevi biter ve kaşif arı olarak kendine rasgele yeni bir kaynak aramaya başlar (Eşitlik 3.1). Burada kullanılan limit değeri algoritma için önemli bir parametredir. Temel ABC algoritmasında her çevrimde sadece bir kaşif arının çıkmasına izin verilir.

3.4. ABC Algoritmasının Sınırlamalı Optimizasyon Problemlerinin Çözümü İçin Uyarlanması

Bu tez çalışmasında sınırlamasız problemlerinin çözümünde kullanılan temel ABC algoritmasına üç farklı yöntem entegre edilerek sınırlamalı problemleri çözebilir hale getirilmiştir.

3.4.1. Ceza Fonksiyonlarını Kullanarak ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü

Bu bölümde sınırlamalı problemleri çözmek için Bölüm 2'de anlatılan ceza fonksiyonlarını kullanarak ABC algoritması (PEN-ABC) geliştirilmiştir. PEN-ABC'deki temel adımlar sınırlamasız problemlerin çözümü için geliştirilen Bölüm 3.3'de gösterilen temel ABC algoritmasındaki [1] ile aynıdır. Ancak ceza fonksiyonlarının kullanımına bağlı olarak içerikte bazı değişiklikler mevcuttur.

Ceza fonksiyonu türlerinden statik ceza yapısı kullanılmıştır. Statik ceza yapısında her problem için verilen sabit bir ceza faktörü bulunmaktadır. Temel ABC algoritmasından farklı olarak her çözüm için hesaplanan ihlal değeri yani sınırlama aşım değeri belli bir eşitlikle amaç fonksiyonuna eklenir. Bölüm 2'de verilen Eşitlik 2.20 ile yeni amaç fonksiyonu oluşturulur. Bu eşitlikte r_g ceza faktörü ve g_i sınırlama aşım miktarıdır. r_g parametresinin farklı değerlerine göre Pen-ABC algoritmasının performansı ölçülmüştür.

Temel ABC algoritmasından farklı olarak Karaboğa ve Akay [39] sınırlamalı optimizasyon problemlerinin çözümünde yakınsama hızını artırmak için komşu çözüm oluştururken değişim oranı (MR) adı verilen bir parametre ve popülasyona kontrollü farklılık (diversity) sağlaması için kaşif üretim periyodu (SPP) adı verilen başka bir parametre eklemiştir. Ayrıca sınırlamalı ABC'de popülasyonda kabul edilebilir bölge dışında çözümler olmasına izin verildiği için gözcü arı aşamasında bu çözümler için olasılık hesabında sınırlama aşım değerlerine göre olasılık atanmaktadır. Farklı olan kısımlar Eşitlik 3.6 ve Eşitlik 3.7 ile gösterilmiştir. Bu eşitlikler haricindeki tüm aşamalar temel ABC algoritması ile aynıdır.

PEN-ABC algoritmasında yeni bir çözüm üretirken Eşitlik 3.6 kullanılır. R_j $[0,1]$ aralığında uniform dağılımlı rasgele sayı, MR $[0,1]$ aralığında değer alan kontrol parametresidir. ϕ_{ij} $[-1,1]$ aralığında uniform dağılımlı rasgele bir sayı ve $k \in \{1,2,...,SN\}$ i'den farklı rasgele bir indistir.

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & if\ R_j < MR \\ x_{ij}, & otherwise \end{cases} \tag{3.6}$$

PEN-ABC algoritmasında gözcü arıların kaynak seçerken kullandıkları olasılık hesabı Eşitlik 3.7'de verilmiştir. $\vec{x}_i$ çözümünün, ceza değeri $\vec{v}iolation_i$ ve uygunluk miktarı $\vec{f}itness_i$'dir.

$$p_i = \begin{cases} 0.5 + \left(\frac{fitness_i}{\sum_{i=1}^{SN} fitness_i}\right) * 0.5 & kabul\ edilebilir\ ise \\ \left(1 - \frac{violation_i}{\sum_{i=1}^{SN} violation_i}\right) * 0.5 & kabul\ edilebilir\ degilse \end{cases} \tag{3.7}$$

3.4.2. Stokastik Beslemeli ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü

Bu yöntem stokastik sıralama (SR) baz alınarak geliştirilmiştir. Stokastik sıralama mantığı güncellenerek ABC algoritmasına entegre edilmiştir ve bu çalışmada kısaca SR-ABC olarak isimlendirilmiştir. SR mantığına Bölüm 2'de değinilmiştir. Runarsson ve Yao [16], ceza fonksiyonlarında kullanılan r_g ceza faktörünün belirlenme zorluğunu ortadan kaldırmak ve amaç fonksiyonu ile ceza fonksiyonu arasında dengeyi sağlamak için stokastik sıralama mantığını önermiştir. Sıralamayı yaparken ise kabarcık sıralama prosedürünü kullanmışlardır (Bölüm 2.3.1.1). Bu yöntemi ABC algoritmasına entegre ederken yine kabarcık sıralama algoritması kullanılmıştır. Ancak ABC algoritmasının mantığına göre uyarlanmıştır.

SR-ABC algoritmasında görevli ve gözcü arı aşamalarında komşu çözümler temel ABC algoritmasındaki gibi oluşturulur. Burada seleksiyon işlemine geçildiğinde ise aç gözlü seleksiyon işlemi uygulanmaz. Stokastik sıralama prosedürü devreye girer. Sadece görevli arı aşamasına stokastik sıralama prosedürü uygulanırsa gözcü arı aşamasında seleksiyon işlemi için Deb'in kuralları kullanılır yada sadece gözcü arı aşamasına stokastik sıralama prosedürü uygulanırsa görevli arı aşamasında Deb'in kuralları kullanılır. Stokastik sıralama prosedüründe öncelikle SN tane mevcut çözüm ve SN tane yeni oluşturulan komşu çözüm birleştirilir. Yani elde edilen $2 * SN$ çözüm stokastik sıralama prosedürü baz alınarak amaç fonksiyonu değeri ve ihlal değerine göre sıralanır. Bu sıralanan çözümler stokastik sıralama algoritmasındaki gibi sıraya göre alınmaz. İçlerinden bazı çözümler sıralı olarak bazıları ise çeşitlilik olması açısından rasgele seçilir. Böylece bu seçilen SN tane çözümün içinde tamamen kabul edilebilir çözümler bulunmamış olur ve popülasyondaki farklılık (diversity) sağlanır. Seçilen SN çözüm için uygunluk değerleri hesaplanır ve ABC algoritmasının diğer adımları temel ABC algoritmasındaki ile aynı devam eder. Stokastik beslemeli ABC algoritması, Algoritma 3 ile gösterilmiştir.

Algoritma 3.2: SR-ABC algoritmasının stokastik besleme prosedürü

```
Mevcut çözümler dizisi ile komşu çözümler dizisini birlestir
for i = 1 to SN * 2 do
    for j = 1 to SN * 2 do
        if φ_j = φ_{j+1} = 0 then
            if f_j > f_{j+1} then
                Yer değiştir
            end if
        end if
        if φ_j > φ_{j+1} then
            Yer değiştir
        end if
    end for
end for
SN tane çözümün bazılarını sıralı bazılarını rasgele sec
```

Hem işçi arı fazında hem de gözcü arı fazında stokastik yapı kullanılarak oluşturulan stokastik beslemeli ABC algoritmasında yakınsama hızını artırmak amacıyla komşu çözüm üretme aşamasında tek komşu yerine birkaç komşu bilgisinin kullanıldığı bir komşuluk mekanizması kullanılmıştır. Sadece görevli arı aşamasında, sadece gözcü arı aşamasında ya da hem görevli arı hemde gözcü arı aşamalarında komşu çözümler üretilirken Diferansiyel Gelişim algoritmasında mutant vektör oluştururken kullanılan Eşitlik 3.8 kullanılarak problemlerin çözümü gerçekleştirilmiştir. Bu eşitlikte $r_1 \neq r_2 \neq i$ olarak varsayılmaktadır.

$$\upsilon_{ij} = \begin{cases} x_{r_1 j} + F * (x_{ij} - x_{r_2 j}), & if\ R_j < MR \\ x_{ij}, & otherwise \end{cases} \tag{3.8}$$

3.4.3. DEB-SR Yöntemini Kullanarak ABC Algoritması ile Sınırlamalı Optimizasyon Problemlerinin Çözümü

Sınırlamalı optimizasyon problemlerinin çözümü için Karaboğa ve Akay [40] tarafından ABC algoritmasının seleksiyon aşamasında Deb'in kuralları kullanılmıştır. Seleksiyon aşamasında Deb'in kuralları kullanıldığında popülasyona sadece kabul edilebilir çözümler dahil olmuştur. Popülasyona farklılık (diversity) katılması ve kabul edilebilir olmayan çözümlerin de dahil edilebilmesi amacıyla Deb'in kurallarına stokastiklik eklenmiştir. Dolayısıyla stokastiklik ve Deb'in kurallarının bir arada kullanıldığı bir yöntem geliştirilmiştir. Bu yöntem DEB-SR-ABC olarak adlandırılmıştır. DEB-SR-ABC

algoritmasının seleksiyon aşamasında Algoritma 3.3 kullanılır. Bu algoritma ile popülasyona çeşitlilik katılması amaçlanmıştır. Popülasyona sadece kabul edilebilir çözümlerin yerine kabul edilebilir olmayan çözümler P_f katsayısının etkisiyle dahil olur.

DEB-SR-ABC algoritmasının seleksiyon aşamasında kullanılan Algoritma 3.3'te mevcut çözüm ve komşu çözümün her ikisi de kabul edilebilir bölgede ise veya rasgele üretilen random sayı P_f'den küçükse amaç fonksiyonu değerine göre seçim yapılır. Mevcut çözüm kabul edilebilir bölgede değil ancak komşu çözümün kabul edilebilir bölgede olduğu durumda, random sayının P_f'den küçük olması şartıyla komşu çözümün amaç fonksiyonu değeri daha iyiyse komşu çözüm seçilir. Komşu çözümün amaç fonksiyonu değeri daha iyi olmadığı halde sınırlama aşım değeri daha küçükse yine komşu çözüm seçilir. Aksi durumda mevcut çözüm korunur. İki çözüm de kabul edilebilir bölgede değilse sınırlama aşım değeri daha küçük olan seçilir.

Algoritma 3.3: DEB-SR-ABC algoritmasının seleksiyon aşaması

```
if (violation_i ≤ 0 and violation_j ≤ 0) or random < P_f then
    if fitness_j > fitness_i then
        Komşu çözümü seç
    else
        Mevcut çözümü tut.
    end if
else
    if (violation_i > 0 and violation_j ≤ 0) then
        if random < P_f then
            if fitness_j > fitness_i then
                Komşu çözümü seç
            end if
        else
            if violation_j < violation_i then
                Komşu çözümü seç
            else
                Mevcut çözümü tut.
            end if
        end if
    end if
else
    if (violation_i > 0 and violation_j > 0) then
        if violation_j < violation_i then
            Komşu çözümü seç
        else
            Mevcut çözümü tut.
        end if
    end if
end if
```

4. BÖLÜM

BULGULAR

4.1. Deneysel Çalışmalar

Bu bölümde tez çalışmasında kullanılan yeni yöntemlerle ilgili deneysel çalışmalara yer verilmiştir. Öncelikle PEN-ABC, SR-ABC ve DEB-SR-ABC algoritmaları test problemleri aracılığıyla değerlendirilmiştir. Geliştirilen üç yöntem sonuçları literatürdeki Stokastik sıralama (SR) metodu [16], geliştirilmiş stokastik sıralama (ISR) metodu [50], aşırı ceza yaklaşımı (OPA) [50], genetik algoritma (GA) [51], basit çok üyeli evrimsel strateji (SMES) [51], diferansiyel gelişim (DE) [40], parçacık sürü optimizasyonu (PSO) algoritması [52] ve ABC algoritması [40] ile karşılaştırılmıştır.

Algoritma Visual Studio 2017'de $C\#$ programlama dili ile kodlanmıştır. Deneyler Intel (R) Core (TM) $i5-3210M$ 2.5 GHz işlemci, $6GB$ RAM ve Windows 10 $x64$ işletim sistemi bulunan bilgisayar ile yapılmıştır.

4.2. Test Problemleri

Geliştirilen yöntemleri test etmek amacıyla Liang vd. [53] tarafından önerilen 24 sınırlamalı optimizasyon problemi kullanılmıştır. Her test probleminin temel özellikleri Tablo 4.1'de özetlenmiştir. Tablo 4.1'de "D" problemdeki değişken sayısı, "NI" nonlineer eşitsizlik sayısı, "LI" lineer eşitsizlik sayısı, "NE" nonlineer eşitlik sayısı ve "LE" ise lineer eşitlik sayısıdır. "α" aktif sınırlamaların sayısı ve "ρ" kabul edilebilir bölgenin araştırma uzayına oranıdır. $\rho = \frac{|F|}{|S|}$ şeklinde hesaplanır ve burada $|S|$ araştırma uzayı ve $|F|$ ise kabul edilebilir bölgedir. g05, g13, g14, g15, g17, g18, g20, g21, g22, g23 problemlerinin kabul edilebilir bölge oranları açısından çok dar bir alana sahip olduğu

görülmektedir. g20 ve g22 problemlerinin aktif sınırlama sayısının fazla olduğu, g16 ve g18 problemlerinin de nonlineer eşitsizlik sayısının çok olduğu görülmektedir. Test problemleri hakkında detaylı bilgi ise EK-1'de verilmiştir.

Tablo 4.1. Test problemlerinin temel özelliklerinin özeti

Problem	D	Problemin Türü	ρ	LI	NI	LE	NE	α
g01	13	Kuadratik	0.0003%	9	0	0	0	6
g02	20	Nonlineer	99.9973%	0	2	0	0	1
g03	10	Polinomial	0.0026%	0	0	0	1	1
g04	5	Kuadratik	27.0079%	0	6	0	0	2
g05	4	Kübik	0.0000%	2	0	0	3	3
g06	2	Kübik	0.0057%	0	2	0	0	2
g07	10	Kuadratik	0.0000%	3	5	0	0	6
g08	2	Nonlineer	0.8581%	0	2	0	0	0
g09	7	Polinomial	0.5199%	0	4	0	0	2
g10	8	Lineer	0.0020%	3	3	0	0	6
g11	2	Kuadratik	0.0973%	0	0	0	1	1
g12	3	Kuadratik	4.7697%	0	1	0	0	0
g13	5	Nonlineer	0.0000%	0	0	0	3	3
g14	10	Nonlineer	0.0000%	0	0	3	0	3
g15	3	Kuadratik	0.0000%	0	0	1	1	2
g16	5	Nonlineer	0.0204%	4	34	0	0	4
g17	6	Nonlineer	0.0000%	0	0	0	4	4
g18	9	Kuadratik	0.0000%	0	12	0	0	6
g19	15	Nonlineer	33.4761%	0	5	0	0	0
g20	24	Lineer	0.0000%	0	6	2	12	16
g21	7	Lineer	0.0000%	0	1	0	5	6
g22	22	Lineer	0.0000%	0	1	8	11	19
g23	9	Lineer	0.0000%	0	2	3	1	6
g24	2	Lineer	79.6556%	0	2	0	0	2

4.3. Kontrol Parametreleri ve Değerleri

Bu çalışmada [40] numaralı çalışmada önerilen parametre değerleri kullanılmıştır. Dolayısıyla koloni büyüklüğü $SN = 40$, maksimum çevrim sayısı $MCN = 6000$, değişim oranı $MR = 0.8$, D probleme ait parametre sayısı olmak üzere $limit = 0.5 * SN * D$ ve $SPP = 0.5 * SN * D$'dir. Eşitlik sınırlamalarını eşitsizlik sınırlamalarına çevirirken kullanılan $\epsilon = 0.001$ olarak işleme katılmıştır. Her bir deney rasgele popülasyonlarla 30 kez koşulmuştur.

PEN-ABC algoritmasını denerken r_g parametresinin beş farklı değeri için performans

değerlendirmesi yapılmıştır. $r_g = \{1, 5, 10, 20, 50\}$ değerleri incelenmiştir.

SR-ABC algoritmasının performansı üç farklı şekilde değerlendirilmiştir. İlk olarak stokastik yapının işçi ve görevli aşamalarında kullanılması baz alınarak incelenmiştir. Sonrasında ise sınırlamaları aşan parametreler için ceza fonksiyonlarında olduğu gibi ihlal değerleri amaç fonksiyonuna eklenerek performansına bakılmıştır. Bu ekleme işleminde $r_g = 1$ alınmıştır. Son olarak DE mutant vektör oluşturma denklemi kullanılarak yapılan performans incelemesinde ise F ölçekleme faktörü 0.8 olarak kabul edilmiştir.

DEB-SR-ABC algoritmasında ise temel ABC algoritmasındaki parametrelere ek olarak kullanılan P_f parametresi Runarsson [16]'un önerdiği şekilde $P_f = 0.45$ olarak kullanılmıştır.

SR [16], gelişimsel stratejide stokastik sıralama mantığına dayanan bir sınırlama ele alış metotu kullanır. $(30, 200) - ES$ kullanılmıştır ve tüm koşmalar 1750 iterasyondan sonra durdurulmuştur. Böylelikle 200 x $1750 = 350,000$ amaç fonksiyonu değerlendirmesi yapılmıştır. Tüm eşitlik sınırlamalarını eşitsizlik sınırlamalarına çevirirken $|h_j| \in \epsilon$ olmak üzere $\epsilon = 0.0001$ seçilmiştir.

ISR [50], 875 iterasyon boyunca $(60, 400) - ES$ kullanmıştır. Dolayısıyla 400 x $875 = 350,000$ amaç fonksiyonu değerlendirilmiştir. Burada SR'den farklı olarak adım büyüklüğünü ölçeklemek için γ kontrol parametresi kullanılmıştır. Adım büyüklüğündeki azalma 0.85'tir.

OPA [50], ceza yaklaşımına dayanan sınırlama ele alış metotuyla $(60, 400) - ES$ kullanılmıştır. $g12$ hariç bütün koşmalar 875 iterasyon sonra durdurulmuştur. $g12$ ise 87 iterasyonda durdurulmuştur. Fonksiyonun değerlendirme sayısı toplamda 400 x $875 = 350,000$'dir.

GA [51], sınırlama ele alış metotlarından Deb'in kurallarını kullanmıştır. Eşitlik sınırlamalarını eşitsizlik sınırlamalarına çevirirken ϵ değeri dinamik olarak belirlenmiştir. Koloni büyüklüğü 200, maksimum iterasyon sayısı 1200, çaprazlama oranı 0.8 ve değişim oranı 0.6'dır. Toplamda 200 x $1200 = 240,000$ amaç fonksiyonu değerlendirmesi yapılmıştır.

SMES [51], $(100, 300) - ES$ ve sınırlamaları ele almak için Deb'in kurallarını

kullanmıştır. İterasyon sayısı 800'dür. Toplam amaç fonksiyonu değerlendirme sayısı 300 x $800 = 240,000$'dir.

DE algoritmasında [40] sınırlamaları ele almak için Deb'in kuralları kullanılmıştır. Koloni büyüklüğü 40, çaprazlama oranı (CR) 0.9, ölçekleme faktörü (F) 0.5 ve maksimum iterasyon sayısı 6000'dir. Dolayısıyla amaç fonksiyonu değerlendirme sayısı 40 x $6000 = 240,000$'dir.

PSO algoritması [52], sınırlamaları ele alış metotlarından Deb'in kurallarını kullanmıştır. Sürü (popülasyon) büyüklüğü 50 ve iterasyon sayısı 7000'dir. Toplamda amaç fonksiyonu değerlendirme sayısı 50 x $7000 = 350,000$'dir. Sosyal ve bilişsel bileşenlerin her ikisi de 1 olarak alınmıştır. Eylemsizlik ağırlığı (inertia weight) $[0.5, 1]$ aralığında uniform dağılmış rasgele bir reel sayıdır. Eşitlik sınırlamalarını eşitsizlik sınırlamalarına çevirirken $|h_j| \in \epsilon$'de kullanılan $\epsilon = 0.001$'dir.

ABC algoritması [40], sınırlamaları ele alırken Deb'in kurallarını kullanmıştır. Koloni büyüklüğü (SN) 40, değişim oranı (MR) 0.8, maksimum çevrim sayısı (MCN) 6000'dir. D probleme ait parametre sayısı olmak üzere $limit = 0.5 * SN * D$ ve $SPP = 0.5 * SN * D$'dir. Amaç fonksiyonu değerlendirme sayısı 40 x $6000 = 240,000$'dir. Deneyler rasgele popülasyonlarla 30 kez tekrarlanmıştır.

4.4. İstatistiksel Sonuçlar

Tablolarda test fonksiyonları ve optimal değerleri, analiz sonucunda bulunan en iyi, ortalama, en kötü değerleri ve standart sapma değeri mevcuttur. Sadece kabul edilebilir çözümler üzerinde yapılan analizde ise ek olarak bulunan çözümler arasında kabul edilebilir çözümlerin oranını gösteren *K.E. Oranı* bulunmaktadır. Tablolarda yer alan NA kabul edilebilir bölgede çözüm bulunamadığını göstermektedir. Tablolarda ulaşılan en iyi sonuçlar koyu olarak gösterilmiştir.

4.4.1. PEN-ABC Sonuçları

PEN-ABC algoritmasının analiz sonuçları Tablo 4.2 ve Tablo 4.3'te gösterilmiştir. Genel itibariyle tablo incelendiğinde PEN-ABC algoritması r_g'nin belli değerlerinde üç

problemde (g08, g11, g12) küresel minimumu ve onsekiz (g01, g02, g03, g04, g05, g06, g07, g09, g10, g13, g14, g15, g16, g17, g18, g19, g20, g24) problemde ise optimale yakın değerleri bulmuştur. Kalan üç (g21, g22, g23) problemde ise küresel optimumu bulamamıştır. g01 probleminde $r_g = 50$ iken en iyi sonuc elde edilmiştir ve r_g değeri arttıkça sonucun iyileştiği görülmektedir. g02 probleminde $r_g = 10$ iken en iyi sonuç elde edilirken $r_g = 1$ iken diğer r_g değerlerine göre daha kötü sonuç alınmıştır. g03 probleminde $r_g = 50$ iken en iyi sonuca ulaşılırken $r_g = 5$'te diğerlerinden daha kötü sonuç elde edilmiştir. g04 probleminde $r_g = 20$ iken en iyi sonuca, $r_g = 5$ iken en kötü sonuca ulaşılmıştır. g05, g06 ve g07 problemlerinde $r_g = 50$ iken en iyi sonuca, $r_g = 1$ iken en kötü sonuca ulaşılmıştır. g08 probleminde dört farklı r_g değerinde (5, 10, 20, 50) optimal sonuca ulaşılmıştır. g09 probleminde $r_g = 20$ iken en iyi sonuca, $r_g = 1$ iken en kötü sonuca ulaşılmıştır. g10 probleminde optimalden uzak sonuçlar elde edilmiştir. g11 probleminde r_g'nin 2 farklı değerinde (20, 50) küresel en iyi elde edilirken diğer iki değerde (5, 10) optimale yakın değerler bulunmuştur. $r_g = 1$ değerinde diğerlerine göre daha kötü sonuç elde edilmiştir. g12 probleminde tüm r_g değerlerinde küresel minimuma ulaşılmıştır. g13 probleminde $r_g = 1$ değerinde küresel en iyiye en yakın sonuca ulaşılırken artan r_g değerleriyle birlikte kabul edilebilir çözümlerin oranı artamasına rağmen sonuçlar küresel en iyiden uzaklaşmıştır. g14 probleminde r_g değerinin artmasıyla sonuçlar küresel minimuma yaklaşmıştır ve en iyi sonuç $r_g = 50$ değerinde elde edilmiştir. g15 probleminde $r_g = 50$ iken en iyi sonuç alınmıştır ancak küresel minimumdan daha küçük sonuç bulunmuştur ve bulunan bu değerin kabul edilebilir bölgede olduğu görülmüştür. g16 probleminde artan r_g değerleriyle sonuçlar iyileşerek optimale yaklaşmış ve $r_g = 50$ değerinde küresel minimuma en yakın sonuç elde edilmiştir. g17 probleminde $r_g = 20$ iken optimale en yakın sonuca ulaşılmıştır. g18 probleminde r_g değerinin artmasıyla sonuçlar optimale yaklaşmış ve $r_g = 50$ iken küresel minimuma en yakın sonuç elde edilmiştir. g19 probleminde $r_g = 50$ iken küresel minimuma yakın olan en iyi sonuç, $r_g = 10$ iken en kötü sonuç elde edilmiştir. g20 probleminde $r_g = 20$ iken küresel minimuma en çok yaklaşan sonuç, $r_g = 10$ iken ise diğerlerine göre küresel minimumdan en uzak sonuç elde edilmiştir. g21, g22 ve g23 problemlerinde elde edilen sonuçlarda küresel minimuma yaklaşılamamıştır. Son olarak g24 probleminde ise r_g değerinin artmasıyla sonuçlar küresel minimuma yaklaşmıştır ve $r_g = 50$ değerinde en iyi sonuca ulaşılmıştır.

Tablo 4.4 ve Tablo 4.5'te PEN-ABC algoritması için sadece kabul edilebilir çözümler dahil edilerek analiz yapılmıştır ve kabul edilebilir çözüm üretilen koşmaların sayısının tüm koşma sayısına (30) oranı verilmiştir. Dokuz problemde (g01, g02, g03, g07, g08, g09, g16, g19, g24) r_g'nin tüm değerleri için $\%100$ oranıyla kabul edilebilir bölge içerisinde çözümler elde edilmiştir. Altı problemde (g03, g06, g11, g13, g15, g18) farklı r_g değerleriyle ve farklı oranlarda kabul edilebilir çözümler elde edilmiştir. Dokuz problemde (g05, g10, g12, g14, g17, g20, g21, g22, g23) elde edilen çözümler kabul edilebilir değildir. r_g değerinin artmasıyla koşmalardaki kabul edilebilirlik oranının arttığı görülmektedir. Bu da ceza katsayısının artmasının çözümleri kabul edilebilir bölgeye ittiğini göstermektedir.

Tablo 4.2. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g01-g12 test problemleri için istatistiksel sonuçları

Fonksiyon	Optimal		En iyi	Ortalama	En Kötü	Standart Sapma
g01	-15.000	rg=1	-14.414	-12.120	-7.789	1.604
		rg=5	-14.915	-14.604	-14.187	0.173
		rg=10	-14.920	-14.794	-14.539	0.100
		rg=20	-14.969	-14.858	-14.583	0.084
		rg=50	**-14.986**	**-14.948**	**-14.889**	**0.023**
g02	0.803619	rg=1	0.802638	0.787720	0.732281	0.017
		rg=5	0.803602	0.789204	0.729568	0.018
		rg=10	**0.803607**	0.791964	0.749700	0.014
		rg=20	0.803591	0.786921	**0.755164**	0.013
		rg=50	0.803599	**0.793056**	0.752692	**0.012**
g03	1	rg=1	0.8830	0.0082	0.5885	0.212
		rg=5	0.3604	0.0240	0.0222	**0.075**
		rg=10	0.7294	0.1141	0.0006	0.194
		rg=20	0.8244	0.2189	0.0006	0.244
		rg=50	**0.9752**	**0.6737**	**0**	0.236
g04	-30665.539	rg=1	-30135.576	-29267.342	-27988.139	495.305
		rg=5	-30127.391	-29210.471	-26444.085	727.588
		rg=10	-30241.728	**-29388.250**	-27412.052	624.031
		rg=20	**-30466.900**	-29385.895	**-28452.547**	**470.696**
		rg=50	-30459.850	-29337.438	-28133.580	536.312
g05	5126.4981	rg=1	5121.0796	**5189.3601**	**5405.0043**	**76.234**
		rg=5	5129.4071	5305.6252	6101.7621	255.520
		rg=10	5126.8550	5312.7931	6083.7368	270.383
		rg=20	5126.5348	5201.1971	5544.2285	84.440
		rg=50	**5126.5284**	5316.7171	6108.5382	247.537
g06	-6961.81388	rg=1	-7950.26324	**-5840.02313**	-1613.03496	1820.893
		rg=5	-7781.58635	-4984.09977	-1466.75352	1856.376
		rg=10	-7848.13164	-5515.71714	**-1782.18932**	1603.436
		rg=20	-7464.22285	-5149.60416	-1143.62277	1580.294
		rg=50	**-6827.40219**	-5523.45991	-1554.44914	**1223.779**
g07	24.3062091	rg=1	25.4577	31.6738	43.1440	3.649
		rg=5	24.7538	25.5460	27.0371	0.567
		rg=10	24.5334	25.0072	26.0738	0.379
		rg=20	24.4382	24.7678	25.3266	0.237
		rg=50	**24.3659**	**24.5914**	**25.0686**	**0.194**
g08	0.095825	rg=1	0.095823	0.094926	0.091793	0.001
		rg=5	**0.095825**	0.095629	0.092365	0.0006
		rg=10	**0.095825**	0.095783	0.095233	0.0001
		rg=20	**0.095825**	**0.095799**	**0.095366**	**0**
		rg=50	**0.095825**	0.095629	0.089960	0.001
g09	680.6300573	rg=1	680.807282	680.915695	681.150225	0.067
		rg=5	680.653749	680.693319	680.794158	0.026
		rg=10	680.640126	680.655039	680.673953	0.008
		rg=20	**680.633721**	680.648931	680.673052	0.009
		rg=50	680.633748	**680.642515**	**680.652127**	**0.005**
g10	7049.25	rg=1	2101.4695	11098.6654	29724.1306	8206.329
		rg=5	2107.2974	8166.6078	25991.6036	8390.319
		rg=10	2114.8156	9386.3641	**24023.8842**	7780.081
		rg=20	2128.7630	**8098.9257**	28243.7040	7409.291
		rg=50	**2171.1908**	9044.0929	24896.5595	**7291.820**
g11	0.75	rg=1	0.739	0.827	1.003	0.102
		rg=5	0.747	0.830	1	0.115
		rg=10	0.748	0.782	1	0.073
		rg=20	**0.75**	**0.75**	**0.750**	**0.0004**
		rg=50	**0.75**	**0.75**	**0.750**	0.0002
g12	1	rg=1	**1**	0.999	0.999	**0**
		rg=5	**1**	**1**	**1**	**0**
		rg=10	**1**	**1**	**1**	**0**
		rg=20	**1**	**1**	**1**	**0**
		rg=50	**1**	0.999	0.999	**0**

Tablo 4.3. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g13-g24 test problemleri için istatistiksel sonuçları

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma
g13	0.0539498	rg=1	**0.05611**	**0.45962**	1.00006	0.386
		rg=5	0.09534	0.79161	**0.99997**	0.300
		rg=10	0.14488	0.75960	1.00541	0.246
		rg=20	0.40804	0.83561	0.99999	0.194
		rg=50	0.56397	0.88431	0.99999	**0.132**
g14	-47.7648	rg=1	-144.8183	-120.8535	**-45.5190**	20.280
		rg=5	-61.0912	-52.6587	-39.4591	5.535
		rg=10	-53.8851	**-47.1637**	-38.4537	3.192
		rg=20	-50.3314	-45.1440	-41.5038	2.495
		rg=50	**-47.3311**	-43.6186	-39.6932	**2.045**
g15	961.715	rg=1	961.531	962.182	964.752	0.874
		rg=5	961.693	962.004	965.315	0.740
		rg=10	961.695	961.958	964.152	0.511
		rg=20	961.708	961.758	962.046	0.084
		rg=50	**961.712**	**961.724**	**961.923**	**0.037**
g16	-1.9051	rg=1	-1.8931	-1.8777	-1.8471	0.012
		rg=5	-1.9033	-1.8986	-1.8897	0.003
		rg=10	-1.9039	-1.9011	-1.8960	0.002
		rg=20	-1.9046	-1.9032	-1.9013	0.0007
		rg=50	**-1.9049**	**-1.9042**	**-1.9031**	**0.0004**
g17	8853.539	rg=1	8411.095	8605.070	9964.8986	311.963
		rg=5	8764.965	**8988.163**	10013.541	271.597
		rg=10	8810.036	9104.684	10826.702	419.657
		rg=20	**8857.119**	9263.355	12581.852	819.885
		rg=50	8885.622	9074.324	**9675.698**	**197.601**
g18	-0.866025	rg=1	-0.8227	-0.5612	-0.1121	0.161
		rg=5	-0.8451	-0.7711	-0.6618	0.052
		rg=10	-0.8510	-0.8068	-0.6474	0.039
		rg=20	-0.8578	-0.8208	-0.6715	0.042
		rg=50	**-0.8633**	**-0.8467**	**-0.8002**	**0.014**
g19	32.65559	rg=1	33.4452	34.6501	35.4027	0.532
		rg=5	33.4989	**34.1788**	35.4297	**0.452**
		rg=10	33.5709	34.5578	**35.3146**	0.489
		rg=20	33.3237	34.3725	35.4928	0.541
		rg=50	**33.2754**	34.3819	35.3445	0.505
g20	0.204979	rg=1	**0.16361**	**0.23796**	**0.39598**	**0.057**
		rg=5	0.26005	0.44328	1.25580	0.206
		rg=10	0.28390	0.45302	0.94323	0.154
		rg=20	0.25355	0.25355	2.33026	0.437
		rg=50	0.27732	0.84175	3.37812	0.807
g21	193.7245	rg=1	0	22.4075	174.6866	47.081
		rg=5	0	16.4042	136.8517	39.268
		rg=10	0	20.5750	289.8088	57.140
		rg=20	0	39.1424	287.7568	80.721
		rg=50	0	26.6338	289.2951	65.490
g22	236.43097	rg=1	0	117.457	1813.5562	332.852
		rg=5	0	53.659	171.410	52.566
		rg=10	0	67.5899	414.151	103.595
		rg=20	0	74.308	538.103	126.683
		rg=50	0	58.038	341.764	74.471
g23	-400.0551	rg=1	-1244.7820	-292.5886	1240.8881	649.659
		rg=5	-1803.5354	17.2841	2871.1869	1206.129
		rg=10	-1435.0973	735.6651	9006.6728	2060.080
		rg=20	-1435.9703	1152.3421	6590.3145	2379.451
		rg=50	-1049.8544	6855.1020	35496.6151	9266.504
g24	-5.50801	rg=1	-5.49183	-5.40780	-5.31011	0.051
		rg=5	-5.50426	-5.48743	-5.45128	0.014
		rg=10	-5.50565	-5.49525	-5.46667	0.009
		rg=20	-5.50652	-5.50144	-5.49082	0.003
		rg=50	**-5.50789**	**-5.50592**	**-5.50083**	**0.001**

Tablo 4.4. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon	Optimal		En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g01	-15.000	rg=1	-14.414	-12.120	-7.789	1.604	%100
		rg=5	-14.915	-14.604	-14.187	0.173	%100
		rg=10	-14.920	-14.794	-14.539	0.100	%100
		rg=20	-14.969	-14.858	-14.583	0.084	%100
		rg=50	**-14.986**	**-14.948**	**-14.889**	**0.023**	%100
g02	0.803619	rg=1	0.802638	0.787720	0.732281	0.017	%100
		rg=5	0.803602	0.789204	0.729568	0.018	%100
		rg=10	**0.803607**	0.791964	0.749700	0.014	%100
		rg=20	0.803591	0.786921	**0.755164**	0.013	%100
		rg=50	0.803599	**0.793056**	0.752692	**0.012**	%100
g03	1	rg=1	NA	NA	NA	NA	%0
		rg=5	0.0382	0.0077	0	**0.017**	%16.66
		rg=10	0.2633	0.0318	0	0.0748	%43.33
		rg=20	0.7271	0.1889	0	0.225	%50
		rg=50	**0.9753**	**0.6795**	**0**	0.239	%96.66
g04	-30665.539	rg=1	-30093.400	-29069.471	-27752.665	583.551	%100
		rg=5	-30259.520	-29300.429	-28384.684	557.931	%100
		rg=10	-30241.728	**-29388.250**	-27412.052	624.031	%100
		rg=20	**-30466.900**	-29385.895	**-28452.547**	**470.696**	%100
		rg=50	-30459.850	-29337.438	-28133.580	536.312	%100
g05	5126.4981	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g06	-6961.81388	rg=1	-6203.2969	-4338.8127	-1613.0349	1374.907	%40
		rg=5	-6854.6295	-4667.9440	-1693.4942	1504.487	%70
		rg=10	-6551.9392	-4896.4923	-1782.1893	1383.354	%63.33
		rg=20	**-6834.2723**	-5236.5037	-3159.1222	1155.429	%70
		rg=50	-6827.4022	**-5607.8828**	**-3326.7508**	**996.979**	%86.66
g07	24.3062091	rg=1	25.4577	31.6738	43.1440	3.649	%100
		rg=5	24.7538	25.5460	27.0371	0.567	%100
		rg=10	24.5334	25.0072	26.0738	0.379	%100
		rg=20	24.4382	24.7678	25.3266	0.237	%100
		rg=50	**24.3659**	**24.5914**	**25.0686**	**0.194**	%100
g08	0.095825	rg=1	0.095823	0.094926	0.091793	0.001	%100
		rg=5	**0.095825**	0.095629	0.092365	0.0006	%100
		rg=10	**0.095825**	0.095783	0.095233	0.0001	%100
		rg=20	**0.095825**	**0.095799**	**0.095366**	**0**	%100
		rg=50	**0.095825**	0.095629	0.089960	0.001	%100
g09	680.6300573	rg=1	680.807282	680.915695	681.150225	0.067	%100
		rg=5	680.653749	680.693319	680.794158	0.026	%100
		rg=10	680.640126	680.655039	680.673953	0.008	%100
		rg=20	**680.633721**	680.648931	680.673052	0.009	%100
		rg=50	680.633748	**680.642515**	**680.652127**	**0.005**	%100
g10	7049.25	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g11	0.75	rg=1	**0.75**	0.927	1	0.120	%13.33
		rg=5	**0.75**	0.852	1	0.125	%73.33
		rg=10	**0.75**	0.783	1	0.074	%96.66
		rg=20	**0.75**	**0.75**	**0.750**	0.0004	%100
		rg=50	**0.75**	**0.75**	**0.750**	**0.0002**	%100
g12	1	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0

Tablo 4.5. Ceza fonksiyonunu kullanan ABC algoritmasının (PEN-ABC) g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon	Optimal		En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g13	0.0539498	rg=1	0.99994	0.99998	1.00000	**0**	%16.66
		rg=5	**0.09534**	0.91169	**0.99997**	0.241	%60
		rg=10	0.14488	**0.75340**	0.99999	0.269	%66.66
		rg=20	0.40804	0.83561	0.99999	0.194	%100
		rg=50	0.56397	0.88431	0.99999	0.132	%100
g14	-47.7648	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g15	961.715	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	961.716	961.754	961.792	0.053	%6.66
		rg=50	**961.713**	**961.719**	**961.734**	**0.005**	%50
g16	-1.9051	rg=1	-1.8931	-1.8777	-1.8471	0.012	%100
		rg=5	-1.9033	-1.8986	-1.8897	0.003	%100
		rg=10	-1.9039	-1.9011	-1.8960	0.002	%100
		rg=20	-1.9046	-1.9032	-1.9013	0.0007	%100
		rg=50	**-1.9049**	**-1.9042**	**-1.9031**	**0.0004**	%100
g17	8853.539	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g18	-0.866025	rg=1	-0.8227	-0.5770	-0.1121	0.141	%86.66
		rg=5	-0.8451	-0.7711	-0.6618	0.052	%100
		rg=10	-0.8510	-0.8068	-0.6474	0.039	%100
		rg=20	-0.8578	-0.8208	-0.6715	0.042	%100
		rg=50	**-0.8633**	**-0.8467**	**-0.8002**	**0.014**	%100
g19	32.65559	rg=1	33.4452	34.6501	35.4027	0.532	%100
		rg=5	33.4989	**34.1788**	35.4297	**0.452**	%100
		rg=10	33.5709	34.5578	**35.3146**	0.489	%100
		rg=20	33.3237	34.3725	35.4928	0.541	%100
		rg=50	**33.2754**	34.3819	35.3445	0.505	%100
g20	0.204979	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g21	193.7245	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g22	236.43097	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g23	-400.0551	rg=1	NA	NA	NA	NA	%0
		rg=5	NA	NA	NA	NA	%0
		rg=10	NA	NA	NA	NA	%0
		rg=20	NA	NA	NA	NA	%0
		rg=50	NA	NA	NA	NA	%0
g24	-5.50801	rg=1	-5.49183	-5.40780	-5.31011	0.051	%100
		rg=5	-5.50426	-5.48743	-5.45128	0.014	%100
		rg=10	-5.50565	-5.49525	-5.46667	0.009	%100
		rg=20	-5.50652	-5.50144	-5.49082	0.003	%100
		rg=50	**-5.50789**	**-5.50592**	**-5.50083**	**0.001**	%100

4.4.2. SR-ABC Sonuçları

Stokastik beslemeli ABC algoritması dört farklı şekilde incelenmiştir. Sadece işçi arı fazına stokastik yapı entegre edilerek (Employed-SR), sadece gözcü arı fazına stokastik yapı entegre edilerek (Onlooker-SR), hem işçi arı hem de gözcü arı fazına stokastik yapı entegre edilerek (Employed-Onlooker-SR) ve hem işçi arı hem de gözcü arı fazına stokastik yapı entegre edilmesine ek olarak sınırlama aşım değerlerinin de amaç fonksiyonuna eklenme işlemi uygulanarak (Employed-Onlooker-SR/rg) analizler yapılmıştır. Bu analizler Tablo 4.6 ve Tablo 4.7'de gösterilmiştir. Sonuçlar genel itibariyle incelendiğinde optimal değere ulaşan yada optimale en yakın değeri bulan yöntem problemden probleme farklılık göstermektedir. Employed-SR yöntemiyle sekiz problemde (g01, g04, g06, g08, g11, g12, g16, g24) optimal değer, on bir problemde (g02, g03, g05, g07, g09, g10, g14, g15, g17, g18, g19) optimale yakın değer bulunmuştur. Kalan beş problemde (g13, g20, g21, g22, g23) etkün sonuç alınamamştır. Onlooker-SR yöntemiyle beş problemde (g04, g06, g08, g11, g12) optimal değer, on dört problemde (g01, g02, g03, g05, g07, g09, g13, g14, g15, g16, g17, g18, g19, g21) optimale yakın değer alınmıştır. Kalan beş problemde (g10, g20, g21, g22, g23) optimal değere ulaşılamamıştır. Employed-Onlooker-SR yöntemiyle altı problemde (g04, g08, g11, g12, g16, g24) optimal değer, on iki problemde (g01, g02, g03, g05, g07, g09, g10, g14, g15, g17, g18, g19) optimale yakın değer bulunmuştur. Kalan altı problemde (g06, g13, g20, g21, g22, g23) ise elde edilen sonuç optimalden uzaktır. Employed-Onlooker-SR/rg yöntemiyle altı problemde (g04, g08, g11, g12, g16, g24) optimal değer, on iki problemde (g01, g02, g03, g05, g07, g09, g10, g14, g15, g17, g18, g19) optimale yakın değer bulunmuştur. Kalan altı problemde (g06, g13, g20, g21, g22, g23) ise küresel minimumdan uzak sonuçlar alınmıştır. Problem bazında sonuçlar ayrı ayrı olarak değerlendirilmiş ve hangi SR entegresinde en iyi sonucu verdiği incelenmiştir. g01 probleminde Employed-SR, g02 probleminde Omlooker-SR ve g03 probleminde Employed-Onlooker-SR veya Employed-Onlooker-SR/rg entegre edilmesiyle en iyi sonuç alınmıştır. g04 probleminde dört yöntemin her birinin entegre edilmesiyle optimal çözüme ulaşılmıştır. g05 probleminde optimal sonuç bulunamamış ancak dört yöntemin her birinin entegre

edilmesi ile optimale yakın sonuçlar elde edilmiştir. g06 probleminde Employed-SR ve Onlooker-SR yöntemleri ile optimal değer elde edilmiştir. g07 probleminde en iyi sonuç Employed-Onlooker-SR veya Employed-Onlooker-SR/rg entegre edilmesi ile alınmıştır. g08 probleminde dört yöntemle optimal değere ulaşılmıştır. g09 probleminde dört yöntemle de optimale yakın sonuç elde edilmiştir. g10 probleminde optimale en yakın sonuç Employed-Onlooker-SR/rg entegre edilmesiyle elde edilmiştir. g11 ve g12 problemlerinde dört yöntemle de optimal sonuçlar elde edilmiştir. g13 probleminde Onlooker-SR entegre edilmesiyle optimale en yakın sonuç elde edilmiştir. g14 probleminde dört yöntemin herhangi biri entegre edilerek optimale yakın sonuç alınmıştır. g15 probleminde dört yöntemle de optimale en yakın sonuç alınmıştır ancak Employed-Onlooker-SR entegre edilerek bulunan sonuç optimalden daha küçüktür. g16 probleminde Employed-SR, Employed-Onlooker-SR veya Employed-Onlooker-SR/rg entegre edilmesiyle optimal çözüm elde edilmiştir, Onlooker-SR entegre edilerek optimale yakın sonuç alınmşıtır. g17 probleminde Employed-Onlooker-SR/rg entegre edilmesiyle optimale en yakın sonuç alınmıştır ancak elde edilen sonuç optimalden daha küçüktür. g18 probleminde Employed-Onlooker-SR/rg entegre edilmesiyle optimale en yakın sonuç elde edilmiştir ancak her bir yöntem optimale yakın değerler bulmaktadır. g19 probleminde Employed-SR entegresiyle optimale en yakın sonuç alınmıştır. g20 probleminde Employed-Onlooker-SR/rg entegre edilmesiyle optimale en yakın sonuç elde edilmiştir. g21 probleminde elde edilen sonuçlardan optimale en yakını Onlooker-SR entegre edilmesiyle olmuştur. g22 ve g23 problemlerinde bu dört yöntemle optimal değere yakın sonuçlar alınamamıştır. g24 probleminde ise tüm yöntemlerden herhangi birinin entegre edilmesiyle küresel en iyi sonuca ulaşılmıştır.

Tablo 4.8 ve Tablo 4.9'da SR-ABC algoritması için sadece kabul edilebilir çözümlere göre değerlendirme yapılmış ve sonuçlar gösterilmiştir. On bir problemde (g01, g02, g03, g04, g07, g08, g09, g11, g16, g19 ve g24) SR'nin tüm uygulamalarında çözümlerin kabul edilebilir bölgede bulunma oranı $\%100$'dür. Altı problemde (g06, g10, g14, g15, g18, g23) çözümler kabul edilebilir bölge içinde yer almaktadır ancak bölgedeki oranı SR'nin uygulamalarına göre fark göstermektedir. Üç problemde (g05, g13, g17) Employed-SR, Employed-Onlooker-SR ve Employed-Onlooker-SR/rg uygulamalarında çözümler farklı oranla kabul edilebilir bölgede yer alırken sadece Onlooker-SR yöntemi

uygulandığında kabul edilebilir çözümler bulunamamıştır. g21 probleminde sadece Employed-SR yönteminin entegre edilmesiyle $\%10$ oranında kabul edilebilir çözüm elde edilmiştir. Geriye kalan üç problemde (g12, g20, g22) bu yöntemde de kabul edilebilir çözümler bulunamamıştır.

Sonuç olarak dört yöntemle de genellikle kabul edilebilirlik oranı yüksek değerler elde edildiği söylenebilir.

Tablo 4.6. Stokastik beslemeli ABC algoritmasının g01-g12 test problemleri için istatistiksel sonuçları

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma
g01	-15.000	Employed-SR	**-15.000**	**-13.766**	**-9.000**	**1.546**
		Onlooker-SR	-14.999	-13.099	**-9.000**	1.729
		Employed-Onlooker-SR	-14.999	-12.679	-7.000	2.139
		Employed-Onlooker-SR/rg	-14.999	-11.916	-6	2.469
g02	0.803619	Employed-SR	0.781265	**0.721954**	**0.489593**	**0.077**
		Onlooker-SR	**0.794766**	0.717791	0.415118	0.098
		Employed-Onlooker-SR	0.788303	0.623135	0.367096	0.128
		Employed-Onlooker-SR/rg	0.791442	0.621560	0.298681	0.127
g03	1	Employed-SR	0,9960	**0,7576**	**0**	**0,336**
		Onlooker-SR	0,9611	0,6143	**0**	0,359
		Employed-Onlooker-SR	**1.0032**	0.4954	**0**	0.480
		Employed-Onlooker-SR/rg	**0.9998**	0.6898	**0**	0.461
g04	-30665.539	Employed-SR	**-30665.539**	-30635.714	-29770.816	163.353
		Onlooker-SR	**-30665.539**	**-30665.539**	**-30665.536**	**0.0007**
		Employed-Onlooker-SR	**-30665.539**	-30653.502	-30308.232	65.212
		Employed-Onlooker-SR/rg	**-30665.539**	-30662.664	-30581.333	15.362
g05	5126.4981	Employed-SR	5133.7862	5495.9674	**6111.0804**	374.009
		Onlooker-SR	5127.7953	5913.8439	6112.6564	**355.438**
		Employed-Onlooker-SR	5128.4927	**5451.8123**	6282.0523	388.108
		Employed-Onlooker-SR/rg	**5127.0429**	5616.4044	10255.2414	938.696
g06	-6961.81388	Employed-SR	**-6961.81388**	**-6961.81388**	-6961.81386	**0**
		Onlooker-SR	**-6961.81388**	**-6961.81388**	**-6961.81387**	**0**
		Employed-Onlooker-SR	-7950.96189	-6992.82073	-6902.87685	181.283
		Employed-Onlooker-SR/rg	-7950.20587	-7027.66865	-6961.18052	250.773
g07	24.3062091	Employed-SR	24.6819	29.3491	122.3473	17.637
		Onlooker-SR	25.1593	**27.2965**	**43.9846**	**3.393**
		Employed-Onlooker-SR	**24.6205**	57.0992	515.8185	92.996
		Employed-Onlooker-SR/rg	24.7066	98.4280	673.4818	165.620
g08	0.095825	Employed-SR	**0.095825**	**0.095825**	**0.095825**	**0**
		Onlooker-SR	**0.095825**	**0.095825**	**0.095825**	**0**
		Employed-Onlooker-SR	**0.095825**	**0.095825**	**0.095825**	**0**
		Employed-Onlooker-SR/rg	**0.095825**	**0.095825**	**0.095825**	**0**
g09	680.6300573	Employed-SR	**680.655688**	**680.755957**	**680.916517**	**0.064**
		Onlooker-SR	680.677814	680.958463	681.401119	0.183
		Employed-Onlooker-SR	680.680946	681.416136	686.023872	0.996
		Employed-Onlooker-SR/rg	680.655841	681.630597	686.459204	1.309
g10	7049.25	Employed-SR	7146.4482	**7820.7732**	**13389.5759**	**1143.997**
		Onlooker-SR	7931.9006	10464.9951	22967.5322	3059.162
		Employed-Onlooker-SR	7113.3981	10395.2964	20588.3659	3798.609
		Employed-Onlooker-SR/rg	**7062.9246**	10620.7167	22903.3216	3942.388
g11	0.75	Employed-SR	**0.75**	**0.75**	**0.75**	**0**
		Onlooker-SR	**0.75**	0.753	0.872	0,022
		Employed-Onlooker-SR	**0.75**	0.832	1	0.119
		Employed-Onlooker-SR/rg	**0.75**	0.776	1	0.076
g12	1	Employed-SR	**1**	**1**	**1**	**0**
		Onlooker-SR	**1**	**1**	**1**	**0**
		Employed-Onlooker-SR	**1**	0.999	0.999	**0**
		Employed-Onlooker-SR/rg	**1**	0.999	0.999	**0**

Tablo 4.7. Stokastik beslemeli ABC algoritmasının g13-g24 test problemleri için istatistiksel sonuçları

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma
g13	0.0539498	Employed-SR	0.27329	1.24207	15.69640	2.736
		Onlooker-SR	**0.06732**	**0.69821**	**1.06046**	**0.271**
		Employed-Onlooker-SR	0.38624	0.92376	2.33476	0.336
		Employed-Onlooker-SR/rg	0.56700	1.04602	5.57209	0.864
g14	-47.7648	Employed-SR	-46.4236	-42.4377	**-37.4757**	**2.120**
		Onlooker-SR	**-47.5973**	**-42.7713**	-36.9997	3.410
		Employed-Onlooker-SR	-47.1281	-41.4926	-37.3170	2.789
		Employed-Onlooker-SR/rg	-46.4469	-40.6021	-34.9307	2.612
g15	961.715	Employed-SR	961.731	965.191	**971.012**	**2.499**
		Onlooker-SR	961.755	966.123	972.317	3.086
		Employed-Onlooker-SR	**961.713**	**964.717**	972.254	3.210
		Employed-Onlooker-SR/rg	961.750	965.603	972.316	3.413
g16	-1.9051	Employed-SR	**-1.9051**	**-1.9049**	**-1.9017**	0.0008
		Onlooker-SR	-1.9050	-1.9046	**-1.9017**	**0.0006**
		Employed-Onlooker-SR	**-1.9051**	-1.8978	-1.7813	0.022
		Employed-Onlooker-SR/rg	**-1.9051**	-1.8565	-1.4306	0.144
g17	8853.539	Employed-SR	8878.154	9088.472	**9312.855**	142.077
		Onlooker-SR	8761,627	9080,607	9322,190	175,213
		Employed-Onlooker-SR	8727.262	9019.858	9316.557	185.864
		Employed-Onlooker-SR/rg	**8834.214**	**9006.661**	9312.954	**131.385**
g18	-0.866025	Employed-SR	-0.8655	**-0.8549**	**-0.6687**	**0.035**
		Onlooker-SR	-0.8340	-0.5048	1.5213	0.445
		Employed-Onlooker-SR	-0.8650	-0.7261	-0.4744	0.148
		Employed-Onlooker-SR/rg	**-0.8657**	-0.7525	-0.4963	0.130
g19	32.65559	Employed-SR	**33.0268**	**36.3225**	**62.6802**	**6.039**
		Onlooker-SR	33.6202	52.9037	417.287	69.122
		Employed-Onlooker-SR	33.8380	70.2747	158.4725	32.638
		Employed-Onlooker-SR/rg	34.3753	70.8285	460.3599	75.841
g20	0.204979	Employed-SR	0.09530	0.16346	**0.27431**	**0.045**
		Onlooker-SR	0.07751	0.14815	0.27351	0.050
		Employed-Onlooker-SR	0.05566	**0.26837**	2.60065	0.452
		Employed-Onlooker-SR/rg	**0.24818**	NA	1.34271	NA
g21	193.7245	Employed-SR	155.587	**771.187**	1000	302.070
		Onlooker-SR	**184.3015**	821.0941	1000	267.561
		Employed-Onlooker-SR	0.0166	852.575	1000	245.105
		Employed-Onlooker-SR/rg	236.8157	891.7951	1000.1414	**212.550**
g22	236.43097	Employed-SR	0	15894.631	20000	6978.546
		Onlooker-SR	0	14685.661	20000	8036.597
		Employed-Onlooker-SR	0	9917.922	20000	8496.266
		Employed-Onlooker-SR/rg	0	1373.999	20000	4135.614
g23	-400.0551	Employed-SR	-1480.4391	-335.5691	543.4071	585.078
		Onlooker-SR	-1379.4954	-315.8909	852.7114	597.926
		Employed-Onlooker-SR	-1884.0801	-435.5528	898.7260	591.239
		Employed-Onlooker-SR/rg	-1701.9959	-502.0586	476.9987	659.872
g24	-5.50801	Employed-SR	**-5.50801**	**-5.50801**	**-5.50801**	**0**
		Onlooker-SR	**-5.50801**	**-5.50801**	**-5.50801**	**0**
		Employed-Onlooker-SR	**-5.50801**	-5.50795	-5.50707	0.0002
		Employed-Onlooker-SR/rg	**-5.50801**	-5.50675	-5.47020	0.006

Tablo 4.8. Stokastik beslemeli ABC algoritmasının g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g01	-15.000	Employed-SR	**-15.000**	**-13.766**	**-9.000**	**1.546**	%100
		Onlooker-SR	-14.999	-13.099	-9.000	1.729	%100
		Employed-Onlooker-SR	-14.999	-12.679	-7	2.139	%100
		Employed-Onlooker-SR/rg	-14.999	-11.916	-6	2.469	%100
g02	0.803619	Employed-SR	0.781265	**0.721954**	**0.489593**	**0.077**	%100
		Onlooker-SR	**0.794766**	0.717791	0.415118	0.098	%100
		Employed-Onlooker-SR	0.788303	0.623135	0.367096	0.128	%100
		Employed-Onlooker-SR/rg	0.791442	0.621560	0.298681	0.127	%100
g03	1	Employed-SR	0.9960	**0.7576**	**0**	**0.336**	%100
		Onlooker-SR	0.9611	0.6143	**0**	0.359	%100
		Employed-Onlooker-SR	**1.0032**	0.4954	**0**	0.480	%100
		Employed-Onlooker-SR/rg	**0.9998**	0.6898	**0**	0.461	%100
g04	-30665.539	Employed-SR	**-30665.539**	-30635.714	-29770.816	163.353	%100
		Onlooker-SR	**-30665.539**	**-30665.539**	**-30665.536**	**0.0007**	%100
		Employed-Onlooker-SR	**-30665.539**	-30653.502	-30308.232	65.212	%100
		Employed-Onlooker-SR/rg	**-30665.539**	-30662.664	-30581.333	15.362	%100
g05	5126.4981	Employed-SR	5133.7862	5545.4614	**6111.0804**	**391.898**	%83.33
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	5128.4927	**5475.6812**	6112.2167	402.432	%66.66
		Employed-Onlooker-SR/rg	**5127.0429**	5591.6518	6112.2248	403.231	%53.33
g06	-6961.81388	Employed-SR	**-6961.81388**	**-6961.81388**	-6961.81386	0	%100
		Onlooker-SR	**-6961.81388**	**-6961.81388**	**-6961.81387**	**0**	%100
		Employed-Onlooker-SR	**-6961.8138**	-6959.7813	-6902.8768	10.944	%96.66
		Employed-Onlooker-SR/rg	**-6961.8138**	-6961.7731	-6961.1805	0.140	%93.33
g07	24.3062091	Employed-SR	24.6819	29.3491	122.3473	17.637	%100
		Onlooker-SR	25.1593	**27.2965**	**43.9846**	**3.393**	%100
		Employed-Onlooker-SR	**24.6205**	57.0992	515.8185	92.996	%100
		Employed-Onlooker-SR/rg	24.7066	98.4280	673.481	165.620	%100
g08	0.095825	Employed-SR	**0.095825**	**0.095825**	**0.095825**	**0**	%100
		Onlooker-SR	**0.095825**	**0.095825**	**0.095825**	**0**	%100
		Employed-Onlooker-SR	**0.095825**	**0.095825**	**0.095825**	**0**	%100
		Employed-Onlooker-SR/rg	**0.095825**	**0.095825**	**0.095825**	**0**	%100
g09	680.6300573	Employed-SR	**680.655688**	**680.755957**	**680.916517**	**0.064**	%100
		Onlooker-SR	680.677814	680.958463	681.401119	0.183	%100
		Employed-Onlooker-SR	680.680946	681.416136	686.023872	0.996	%100
		Employed-Onlooker-SR/rg	680.655841	681.630597	686.459204	1.309	%100
g10	7049.25	Employed-SR	7146.4482	**7820.7732**	13389.5759	1143.997	%100
		Onlooker-SR	7931.9006	8626.0429	**9593.1027**	**513.677**	%56.66
		Employed-Onlooker-SR	7113.3981	9563.1814	20100	2792.186	%86.66
		Employed-Onlooker-SR/rg	**7062.9246**	10416.0424	22903.3216	3952.257	%93.33
g11	0.75	Employed-SR	**0.75**	**0.75**	**0.75**	**0**	%100
		Onlooker-SR	**0.75**	0.753	0.872	0,022	%100
		Employed-Onlooker-SR	**0.75**	0.832	1	0.119	%100
		Employed-Onlooker-SR/rg	**0.75**	0.776	1	0.076	%100
g12	1	Employed-SR	NA	NA	NA	NA	%0
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR/rg	NA	NA	NA	NA	%0

Tablo 4.9. Stokastik beslemeli ABC algoritmasının g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g13	0.0539498	Employed-SR	0.27329	1.24207	15.69640	2.736	%100
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	**0.38624**	**0.92113**	**2.33476**	**0.342**	%96.66
		Employed-Onlooker-SR/rg	0.56700	1.04602	5.57209	0.864	%100
g14	-47.7648	Employed-SR	-46.4236	-42.4377	-37.4757	**2.120**	%100
		Onlooker-SR	**-47.5973**	**-44.5944**	**-38.2042**	3.058	%50
		Employed-Onlooker-SR	-47.1281	-41.4926	-37.3170	2.789	%100
		Employed-Onlooker-SR/rg	-46.4469	-40.6021	-34.9307	2.612	%100
g15	961.715	Employed-SR	961.731	965.191	**971.012**	**2.499**	%100
		Onlooker-SR	961.810	967.399	972.288	3.226	%36.66
		Employed-Onlooker-SR	**961.713**	**964.717**	972.254	3.210	%100
		Employed-Onlooker-SR/rg	961.750	965.603	972.316	3.413	%100
g16	-1.9051	Employed-SR	**-1.9051**	**-1.9049**	**-1.9017**	0.0008	%100
		Onlooker-SR	-1.9050	-1.9046	**-1.9017**	**0.0006**	%100
		Employed-Onlooker-SR	**-1.9051**	-1.8978	-1.7813	0.022	%100
		Employed-Onlooker-SR/rg	**-1.9051**	-1.8565	-1.4306	0.144	%100
g17	8853.539	Employed-SR	8878.154	9088.472	9312.855	142.077	%100
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	8885.724	9058.293	9312.858	148.754	%56.66
		Employed-Onlooker-SR/rg	**8867.080**	**9008.610**	**9215.392**	**103.741**	%76.66
g18	-0.866025	Employed-SR	-0.8655	**-0.8549**	**-0.6687**	**0.035**	%100
		Onlooker-SR	-0.8340	-0.6154	-0.1765	0.181	%90
		Employed-Onlooker-SR	-0.8650	-0.7261	-0.4744	0.148	%100
		Employed-Onlooker-SR/rg	**-0.8657**	-0.7525	-0.4963	0.130	%100
g19	32.65559	Employed-SR	**33.0268**	**36.3225**	**62.6802**	**6.039**	%100
		Onlooker-SR	33.6202	52.9037	417.287	69.122	%100
		Employed-Onlooker-SR	33.8380	70.2747	158.4725	32.638	%100
		Employed-Onlooker-SR/rg	34.3753	70.8285	460.3599	75.841	%100
g20	0.204979	Employed-SR	NA	NA	NA	NA	%0
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR/rg	NA	NA	NA	NA	%0
g21	193.7245	Employed-SR	**320.1101**	**320.1101**	**320.1101**	**0.001**	%10
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR/rg	NA	NA	NA	NA	%0
g22	236.43097	Employed-SR	NA	NA	NA	NA	%0
		Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR	NA	NA	NA	NA	%0
		Employed-Onlooker-SR/rg	NA	NA	NA	NA	%0
g23	-400.0551	Employed-SR	-98.6722	244.1981	543.4071	244.968	%30
		Onlooker-SR	**-149.1551**	296.5747	852.7114	360.462	%30
		Employed-Onlooker-SR	-749.5795	**123.9590**	898.7260	601.201	%16.66
		Employed-Onlooker-SR/rg	290.2389	351.2597	**375.3291**	**34.817**	%16.66
g24	-5.50801	Employed-SR	**-5.50801**	**-5.50801**	**-5.50801**	**0**	%100
		Onlooker-SR	**-5.50801**	**-5.50801**	**-5.50801**	**0**	%100
		Employed-Onlooker-SR	**-5.50801**	-5.50795	-5.50707	0.0002	%100
		Employed-Onlooker-SR/rg	**-5.50801**	-5.50675	-5.47020	0.006	%100

4.4.3. SR-ABC-DE Sonuçları

Tablo 4.10 ve Tablo 4.11 ABC algoritmasının işçi ve gözcü arı fazlarında stokastik yapı kullanılmasına ek olarak komşu çözüm üretme aşamasında DE algoritmasının komşu üretme mekanizması kullanılarak ortaya çıkan sonuçları göstermektedir. Burada DE komşu üretme mekanizması sadece işçi arı fazında kullanıldığında (Employed-DE), sadece gözcü arı fazında kullanıldığında (Onlooker-DE) ve hem işçi arı hem de gözcü arı fazında kullanıldığında (Employed-onlooker-DE) sonuçlar incelenmiştir. Employed-DE uygulandığında iki problemde (g08, g11) optimal değer ve on problemde (g01, g04, g05, g09, g12, g15, g16, g18, g19, g24) optimale yakın değer bulunmuştur. Onlooker-DE uygulandığında iki problemde (g08, g11) optimal değer ve dokuz problemde (g01, g04, g05, g09, g12, g15, g16, g18, g24) optimale yakın değer bulunmuştur. Employed-onlooker-DE uygulandığında ise iki problemde (g08, g11) optimal değer ve yedi problemde (g01, g04, g12, g14, g15, g16, g24) optimale yakın değer bulunmuştur. Problem bazında hangi uygulamada en iyi sonuçlar alındığına bakılarak performans değerlendirmesi yapılabilir. g01 probleminde denenen üç yöntemde de küresel minimuma çok yakın sonuçlar elde edilmiştir ve bu sonuçlardan en iyisi Onlooker-DE uygulanmasıyla elde edilendir. g02 probleminde küresel minimum bulunamamıştır ancak Onlooker-DE uygulanmasıyla diğer uygulamalardan daha iyi sonuç elde edilmiştir. g03 probleminde küresel minimum bulunamamış ancak en yakın sonuç Employed-DE uygulanmasıyla elde edilmiştir. g04 probleminde en iyi sonuca Onlooker-DE uygulanmasıyla ulaşılmasına rağmen ortalama ve standart sapma değerlerine bakıldığında Employed-onlooker-DE uygulandığında daha iyi sonuç aldığı görülmektedir. g05 probleminde üç uygulamada da optimale yakın sonuçlar elde edilmiştir ve bu sonuçlardan optimal değere en yakını Employed-DE uygulanması ile elde edilendir. g06 ve g07 problemlerinde üç uygulamada da optimal değerden uzak sonuçlar elde edilmiştir. g08 problemimde üç uygulamada da küresel minimuma ulaşılmıştır. g09 probleminde üç uygulamada da küresel minimuma yakın sonuç bulunmuş ancak Onlooker-DE uygulamasında elde edilen sonuç diğerlerinden daha iyidir. g10 probleminde üç uygulamada da küresel minimumdan uzakta sonuçlar elde edilmiştir. g11 ve g12 problemlerinde üç uygulamada da küresel minimuma ulaşılmıştır. g13 probleminde üç uygulamada da küresel minimuma yakın sonuçlar alınamamıştır.

g14 probleminde ise Employed-DE ve Employed-onlooker-DE uygulamalarında küresel minimuma yakın sonuç elde edilmiştir. g15 probleminde üç uygulamada küresel minimuma yakın sonuçlar elde edilmiştir. g16 probleminde üç uygulamada da küresel minimuma yakın sonuçlar elde edilmiştir ancak bu sonuçlardan en iyisi Employed-DE uygulaması ile alınan sonuçtur. g17 ve g18 problemlerinde üç uygulamada benzer sonuçlarla küresel minimumun uzağında sonuçlar elde edilmiştir. g19 probleminde küresel minimuma yakın sonuçlar alınsa da üç uygulamada da tam olarak en iyi değere ulaşılamamıştır. g20, g21, g22 ve g23 problemlerinde üç uygulamada da küresel minimumdan uzak sonuçlar elde edilmiştir. Son olarak g24 probleminde Employed-DE uygulamasında küresel en iyiye en yakın sonuç edilmiştir.

Tablo 4.12 ve Tablo 4.13 DE mutant vektör oluşturma eşitliği kullanılarak SR-ABC'nin sadece kabul edilebilir çözümler için istatistiksel sonuçlarını göstermektedir. Dokuz problemde (g01, g02, g03, g04, g08, g09, g11, g19, g24) elde edilen çözümler $\%100$ oranla kabul edilebilir bölge içerisinde yer almaktadır. Yedi problemde (g05, g12, g13, g17, g20, g21, g22) elde edilen çözümler tamamen kabul edilebilir bölge dışındadır. Kalan sekiz problemde ise uygulanan yönteme göre kabul edilebilir çözümlerin oranı değişiklik göstermektedir. g06 probleminde çözümler genel olarak $\%70$'in üzerinde oranla kabul edilebilir bölgede bulunurken, Onlooker-DE uygulamasında bu oran $\%96.66$'yı bulmaktadır. g07 probleminde ise Employed-SR uygulamasında oran $\%96.66$ iken Onlooker-DE ve Employed-onlooker-DE uygulamalarında bu oran $\%100$'dür. g10 probleminde elde edilen çözümlerin en düşük kabul edilebilirlik oranı $\%80$ (Employed-onlooker-DE), en yüksek kabul edilebilirlik oranı $\%96.66$ (Employed-DE, Onlooker-DE)'dır. g14 ve g23 problemlerinde elde edilen çözümlerin ortalama oranı yaklaşık $\%50$'dir. g15 probleminin kabul edilebilir bölgesi çok küçük bir alan olduğundan dolayı çözümler çok düşük oranda ($\%10$) bu bölgede bulunurlar. g16 probleminde bulunan çözümler Employed-onlooker-DE uygulamasında $\%90$ oranıyla kabul edilebilir bölge içerisindeyken, Employed-DE ve Onlooker-DE uygulamasında $\%100$ oranla kabul edilebilir bölgede yer alır. g18 problemi Employed-DE ($\%100$) ve Onlooker-DE ($\%96.66$) uygulamalarında yüksek oranda kabul edilebilir çözüm bulurken Employed-onlooker-DE ($\%56.66$) uygulamasında bu oran düşmektedir.

Tablo 4.10. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğnii kullanarak g01-g12 test problemleri için istatistiksel sonuçları

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma
g01	-15.000	Employed-DE (SR)	-14.988	-10.399	-6	2.059
		Onlooker-DE(SR)	**-14.999**	**-11.631**	**-7.999**	**2.041**
		Employed-onlooker-DE(SR)	-14.971	-10.719	-6	2.190
g02	0.803619	Employed-DE (SR)	0.599782	0.466272	**0.295742**	**0.083**
		Onlooker-DE(SR)	**0.691965**	**0.487656**	0.266249	0.131
		Employed-onlooker-DE(SR)	0.617807	0.432700	0.287479	0.088
g03	1	Employed-DE (SR)	0.5547	0.0525	0	**0.135**
		Onlooker-DE(SR)	**0.6394**	**0.0734**	**0**	0.192
		Employed-onlooker-DE(SR)	-	-	-	-
g04	-30665.539	Employed-DE (SR)	-30665.291	-30632.953	-29770.594	162.884
		Onlooker-DE(SR)	**-30665.480**	-30588.714	-29770.809	238.817
		Employed-onlooker-DE(SR)	-30665.006	**-30661.255**	**-30655.013**	**2.77**
g05	5126.4981	Employed-DE (SR)	**5129.6687**	5580.9178	6282.3207	**413.288**
		Onlooker-DE(SR)	5139.1321	**5656.9637**	**6121.8423**	431.177
		Employed-onlooker-DE(SR)	5167.0007	5701.4418	6285.1512	432.444
g06	-6961.81388	Employed-DE (SR)	**-7950.96285**	-6891.76790	-719.78275	1228.106
		Onlooker-DE(SR)	-7973	**-6980.58072**	**-6928.70786**	**187.563**
		Employed-onlooker-DE(SR)	-7950.96369	-6440.20537	-716.93982	1989.359
g07	24.3062091	Employed-DE (SR)	31.9284	**315.3935**	**3254.1764**	**604.913**
		Onlooker-DE(SR)	**30.0470**	483.8817	3604.0548	976.314
		Employed-onlooker-DE(SR)	41.0585	376.6125	3865.5775	692.884
g08	0.095825	Employed-DE (SR)	**0.095825**	0.095824	0.095823	0
		Onlooker-DE(SR)	**0.095825**	**0.095825**	**0.095824**	**0**
		Employed-onlooker-DE(SR)	**0.095825**	0.095817	0.095782	0
g09	680.6300573	Employed-DE (SR)	682.563225	1017.019757	10530.841601	1796.882
		Onlooker-DE(SR)	**681.994293**	**1014.704063**	**10528.27159**	1796.830
		Employed-onlooker-DE(SR)	688.752013	1033.50943	10540.690585	**1795.653**
g10	7049.25	Employed-DE (SR)	**7794.7728**	12152.6088	24023.9882	4835.733
		Onlooker-DE(SR)	8046.9792	**11304.6050**	**23922.5329**	**3992.848**
		Employed-onlooker-DE(SR)	8269.4539	14177.4423	25220.2402	5272.262
g11	0.75	Employed-DE (SR)	**0.75**	0.840	1	0.122
		Onlooker-DE(SR)	**0.75**	**0.824**	**1**	**0.116**
		Employed-onlooker-DE(SR)	**0.75**	0.832	1	0.119
g12	1	Employed-DE (SR)	0,9999	0,9999	0,9999	0
		Onlooker-DE(SR)	0,9999	0,9999	0,9999	0
		Employed-onlooker-DE(SR)	0,9999	0,9999	0,9999	0

Tablo 4.11. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğini kullanarak g13-g24 test problemleri için istatistiksel sonuçları

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma
g13	0.0539498	Employed-DE (SR)	0.50870	1.50486	14.14416	2.450
		Onlooker-DE(SR)	0.46625	1.73945	18.20509	3.208
		Employed-onlooker-DE(SR)	**0.26848**	**1.21924**	**6.59579**	**1.174**
g14	-47.7648	Employed-DE (SR)	-44.6140	-40.5578	**-35.4910**	**2.012**
		Onlooker-DE(SR)	-137.1640	**-43.7267**	-35.3021	17.865
		Employed-onlooker-DE(SR)	**-47.3858**	-39.7414	-33.8681	3.419
g15	961.715	Employed-DE (SR)	958.536	966.027	972.323	4.222
		Onlooker-DE(SR)	**961.713**	**965.716**	**972.315**	**3.307**
		Employed-onlooker-DE(SR)	958.474	966.838	972.323	4.798
g16	-1.9051	Employed-DE (SR)	**-1.9048**	**-1.8835**	**-1.4146**	**0.088**
		Onlooker-DE(SR)	-1.9047	-1.8697	-1.4272	0.119
		Employed-onlooker-DE(SR)	-1.9016	-1.8001	-1.3022	0.186
g17	8853.539	Employed-DE (SR)	**8697.398**	9002.462	9348.433	**181.522**
		Onlooker-DE(SR)	8696.841	9045.805	**9330.209**	196.502
		Employed-onlooker-DE(SR)	8697.123	**8906.333**	9448.503	206.393
g18	-0.866025	Employed-DE (SR)	-0.7252	-0.4805	**-0.3177**	0.143
		Onlooker-DE(SR)	**-0.7486**	**-0.5320**	-0.2314	**0.141**
		Employed-onlooker-DE(SR)	-0.6682	-0.2608	0.0650	0.166
g19	32.65559	Employed-DE (SR)	34.8465	**102.5446**	609.2800	125.419
		Onlooker-DE(SR)	**34.3165**	102.7691	**482.4082**	**105.677**
		Employed-onlooker-DE(SR)	36.7651	170.0933	661.5886	177.991
g20	0.204979	Employed-DE (SR)	0.05693	0.48412	5.5	1.056
		Onlooker-DE(SR)	**0.05837**	**0.43308**	**2.6**	**0.518**
		Employed-onlooker-DE(SR)	0.04407	0.80637	5.5	1.229
g21	193.7245	Employed-DE (SR)	13.0125	**885.5469**	**1000**	264.366
		Onlooker-DE(SR)	13.4510	896.5388	**1000**	253.061
		Employed-onlooker-DE(SR)	**250.4583**	894.3718	**1000**	**194.863**
g22	236.43097	Employed-DE (SR)	14755.493	20000	7782.877	NA
		Onlooker-DE(SR)	12347.717	20000	9054.792	NA
		Employed-onlooker-DE(SR)	13343.977	20000	9136.176	NA
g23	-400.0551	Employed-DE (SR)	-1701.9959	-502.0586	476.9987	659.872
		Onlooker-DE(SR)	-2100	-162.3325	900	751.309
		Employed-onlooker-DE(SR)	-2100.0059	-209.0839	900	763.512
g24	-5.50801	Employed-DE (SR)	**-5.50795**	-5.50529	-5.49901	0.002
		Onlooker-DE(SR)	-5.50792	**-5.50644**	**-5.50251**	**0.001**
		Employed-onlooker-DE(SR)	-5.50701	-5.49998	-5.48600	0.006

Tablo 4.12. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğnii kullanarak g01-g12 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g01	-15.000	Employed-DE (SR)	-14.997	**-10.565**	-5.00	2.526	%100
		Onlooker-DE(SR)	**-14.998**	-10.389	-4	2.648	%100
		Employed-onlooker-DE(SR)	-14,979	-10,227	**-6**	**2,277**	%100
g02	0.803619	Employed-DE (SR)	0.67403	0.48994	0.29838	0.102	%100
		Onlooker-DE(SR)	0.692254	0.509881	0.298711	**0.095**	%100
		Employed-onlooker-DE(SR)	**0,773185**	**0,557026**	**0,333229**	0,126	%100
g03	1	Employed-DE (SR)	**0,9682**	0,9178	0,8603	0,035	%30
		Onlooker-DE(SR)	0,9670	**0,9404**	**0,8946**	**0,027**	%46.66
		Employed-onlooker-DE(SR)	0,8960	0,8481	0,77512	0,052	%13.33
g04	-30665.539	Employed-DE (SR)	-30665.156	-30662.877	-30652.311	2.422	%100
		Onlooker-DE(SR)	-30665.448	**-30664.178**	**-30661.878**	**0.953**	%100
		Employed-onlooker-DE(SR)	**-30665,538**	-30649,559	-30186,158	87,522	%100
g05	5126.4981	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g06	-6961.81388	Employed-DE (SR)	-6950,5445	**-6925,6439**	**-6873,2064**	**15,305**	%86.66
		Onlooker-DE(SR)	-6959,6382	-6882,2971	-5433,3281	308,718	%80
		Employed-onlooker-DE(SR)	**-6961,7915**	-6755,2957	-6248,8815	206,8479	%60
g07	24.3062091	Employed-DE (SR)	31.5545	255.9783	3097.6876	571.631	%100
		Onlooker-DE(SR)	30,8147	579,8242	3254,0202	1056,327	%96.66
		Employed-onlooker-DE(SR)	**24,3062**	**46,8820**	**182,7588**	**52,475**	%100
g08	0.095825	Employed-DE (SR)	**0.095825**	0.095824	0.095822	0	%100
		Onlooker-DE(SR)	**0.095825**	**0.095825**	**0.095825**	**0**	%100
		Employed-onlooker-DE(SR)	**0.095825**	**0.095825**	**0.095825**	**0**	%100
g09	680.6300573	Employed-DE (SR)	683.463636	688.454996	706.3087138	4.594	%100
		Onlooker-DE(SR)	682.334016	687.898887	709.559949	5.596	%100
		Employed-onlooker-DE(SR)	**682.215039**	**686.341505**	**695.314428**	**3.173**	%100
g10	7049.25	Employed-DE (SR)	7979,842	11423,162	25020,3226	4466,557	%93.33
		Onlooker-DE(SR)	7429.8968	11135.7762	**20999.999**	**3653.603**	%100
		Employed-onlooker-DE(SR)	**7049,2491**	**9335,8127**	25274,1066	4544,089	%100
g11	0.75	Employed-DE (SR)	**0.75**	0.8841	1	0.125	%100
		Onlooker-DE(SR)	**0.75**	0.9340	1	**0.111**	%100
		Employed-onlooker-DE(SR)	**0.75**	**0.8499**	**1**	0.124	%100
g12	1	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0

Tablo 4.13. Stokastik beslemeli ABC algoritmasının DE mutant vektör oluşturma eşitliğini kullanarak g13-g24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon / Optimal			En iyi	Ortalama	En Kötü	Standart Sapma	K.E. Oran
g13	0.0539498	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g14	-47.7648	Employed-DE (SR)	-42.6800	-39.7195	-35.4910	1.960	%56.66
		Onlooker-DE(SR)	**-47.4109**	-40.4281	-35.3021	**3.282**	%66.66
		Employed-onlooker-DE(SR)	-47.3858	**-41.2826**	**-38.2175**	3.422	%36.66
g15	961.715	Employed-DE (SR)	962.937	967.591	972.317	**4.690**	%10
		Onlooker-DE(SR)	**961.726**	**967.176**	**972.312**	5.300	%10
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g16	-1.9051	Employed-DE (SR)	-1.9047	-1.8670	-1.3780	0.126	%100
		Onlooker-DE(SR)	**-1.9048**	**-1.8701**	-1.4148	**0.121**	%100
		Employed-onlooker-DE(SR)	-1.9016	-1.8472	**-1.4196**	0.124	%90
g17	8853.539	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g18	-0.866025	Employed-DE (SR)	-0.7252	-0.4805	**-0.3177**	0.143	%100
		Onlooker-DE(SR)	**-0.7486**	**-0.5423**	-0.25406	**0.132**	%96.66
		Employed-onlooker-DE(SR)	-0.6682	-0.3189	-0.0757	0.146	%56.66
g19	32.65559	Employed-DE (SR)	34.8465	**102.5446**	609.2800	125.419	%100
		Onlooker-DE(SR)	**34.3165**	102.7691	**482.4082**	**105.677**	%100
		Employed-onlooker-DE(SR)	36.7651	170.0933	661.5886	177.991	%100
g20	0.204979	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g21	193.7245	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g22	236.43097	Employed-DE (SR)	NA	NA	NA	NA	%0
		Onlooker-DE(SR)	NA	NA	NA	NA	%0
		Employed-onlooker-DE(SR)	NA	NA	NA	NA	%0
g23	-400.0551	Employed-DE (SR)	-0.0299	**171.1393**	900	**288.960**	%56.66
		Onlooker-DE(SR)	**-123.9304**	277.2296	900	390.338	%46.66
		Employed-onlooker-DE(SR)	-0.0149	211.1070	900	359.558	%60
g24	-5.50801	Employed-DE (SR)	**-5.50795**	-5.50529	-5.49901	0.002	%100
		Onlooker-DE(SR)	-5.50792	**-5.50644**	**-5.50251**	**0.001**	%100
		Employed-onlooker-DE(SR)	-5.50701	-5.49998	-5.48600	0.006	%100

4.4.4. DEB-SR-ABC Sonuçları

DEB-SR yöntemiyle alınan sonuçlar Tablo 4.14'te gösterilmiştir. Bu yöntemle yedi problemde (g01, g04, g08, g11, g12, g16, g24) optimal değer ve yedi problemde (g02, g03, g07, g09, g13, g15, g19) optimale yakın değer bulunmuştur. On problemde ise

küresel minimum bulunamamıştır. Tablo 4.15'te ise sadece kabul edilebilir çözümler üzerinden sonuçlar listelenmiştir. Bu tablo incelendiğinde onbir problemde (g01, g02, g03, g04, g07, g08, g09, g11, g16, g19, g24) bulunan çözümlerin $\%100$ oranla kabul edilebilir bölge içerisinde bulunduğu (g06 $\%80$ oranla) ve oniki problemde (g05, g10, g12, g13, g14, g15, g17, g18, g20, g21, g22, g23) ise çözümlerin tamamen kabul edilebilir bölge dışında olduğu görülmektedir.

Tablo 4.14. DEB-SR yöntemiyle ABC algoritmasının 24 test problemi için istatistiksel sonuçları

Fonksiyon / Optimal		En iyi	Ortalama	En Kötü	Standart Sapma
g01	-15.000	**-15,000**	-15,000	-15,000	0
g02	0.803619	0,8035917	0,7916718	0,7703841	0,010
g03	1	1,0049087	1,0025413	0,992502	0,003
g04	-30665.539	**-30665,53867178**	-30665,53867178	-30665,53867178	0
g05	5126.4981	4986,9871223	5559,6410011	6444,503097	494,511
g06	-6961.81388	-7661,00942	-6344,10005	-3571,62647	1063,805
g07	24.3062091	24,343841	24,513237	25,079750	0,190
g08	0.095825	**0,095825041**	0,095825041	0,095825041	0
g09	680.6300573	680,6339968	680,6418870	680,6506023	0,004
g10	7049.25	2100	11091,6829	23333,4204	6517,067
g11	0.75	**0,749**	0,750	0,782	0,006
g12	1	**1**	1	1	0
g13	0.0539498	0,05730112	0,59645941	1,4169003	0,365
g14	-47.7648	-110,07724	-66,25168	-33,29789	21,410
g15	961.715	961,787783	965,605980	972,509623	3,146
g16	-1.9051	**-1,9051552585**	-1,8861091837	-1,5932476737	0,062
g17	8853.539	8595,28840	8745,61822	8967,17965	75,934
g18	-0.866025	-5,92630	-0,776355	3,01260	1,860
g19	32.65559	33,704527	34,567102	36,035472	0,530
g20	0.204979	0,061084	0,091992	0,164298	0,022
g21	193.7245	0	0	0	0
g22	236.43097	0	0	0	0
g23	-400.0551	-2100	-1439,021	184,804	644,047
g24	-5.50801	**-5,50801327159**	-5,50801327159	-5,50801327159	0

Tablo 4.15. DEB-SR yöntemiyle ABC algoritmasının 24 test problemlerinde kabul edilebilir çözümlerin üretildiği koşmaların sayısının tüm koşma sayısına oranı ve kabul edilebilir çözümlerin üretildiği koşmalara ait istatistikler

Fonksiyon / Optimal	En iyi	Ortalama	En Kötü	Standart Sapma	
g01 -15.000	**-15,000**	-15,000	-15,000	0	%100
g02 0.803619	0,8035917	0,7916718	0,7703841	0,010	%100
g03 1	1,0049087	1,0025413	0,992502	0,003	%100
g04 -30665.539	**-30665,53867178**	-30665,53867178	-30665,53867178	0	%100
g05 5126.4981	NA	NA	NA	NA	%0
g06 -6961.81388	**-6961,813876**	-6240,699255	-3571,626479	934,278796	%80
g07 24.3062091	24,343841	24,513237	25,079750	0,190	%100
g08 0.095825	**0,095825041**	0,095825041,	0,095825041	0	%100
g09 680.6300573	680,6339968	680,6418870	680,6506023	0,004	%100
g10 7049.25	NA	NA	NA	NA	%0
g11 0.75	**0,749**	0,750	0,782	0,006	%100
g12 1	**1**	1	1	0	%0
g13 0.0539498	NA	NA	NA	NA	%0
g14 -47.7648	NA	NA	NA	NA	%0
g15 961.715	NA	NA	NA	NA	%0
g16 -1.9051	**-1,9051552585**	-1,8861091837	-1,5932476737	0,062	%100
g17 8853.539	NA	NA	NA	NA	%0
g18 -0.866025	NA	NA	NA	NA	%0
g19 32.65559	33,704527	34,567102	36,035472	0,530	%100
g20 0.204979	NA	NA	NA	NA	%0
g21 193.7245	NA	NA	NA	NA	%0
g22 236.43097	NA	NA	NA	NA	%0
g23 -400.0551	NA	NA	NA	NA	%0
g24 -5.50801	**-5,50801327159**	-5,50801327159	-5,50801327159	0	%100

4.4.5. Önerilen Yöntemlerin Kıyaslanması

Yapılan analizler sonucunda hangi yöntem hangi problemin çözümünde uygulandığında en sonuç alınır bilgisi elde edilir. g01 probleminin çözümü için Employed-SR ve DEB-SR-ABC yöntemleri kullanılabilir. g02 probleminin çözümü için PEN-ABC ($r_g = \{5, 10, 50\}$) ve DEB-SR-ABC yöntemi kullanılabilir. g03 probleminin çözümü için Employed-Onlooker-SR/rg ve DEB-SR-ABC yöntemlerinden faydalanılabilir. g04 problemi Onlooker-SR ve DEB-SR-ABC yöntemleri ile çözülebilir. g05 probleminde PEN-ABC ($r_g = 10, 20, 50$) yöntemiyle optimale yakın sonuç alınabilir. g06 probleminde Onlooker-SR yöntemi ile optimale ulaşılabilir. g07 problemi optimale en yakın değer bulan PEN-ABC ($r_g = 50$) ve DEB-SR-ABC yöntemleri ile çözülebilir. g08 probleminde önerilen tüm yöntemler ile çözüme ulaşılabilir. g09 probleminin çözümünde PEN-ABC ($r_g = \{20, 50\}$), Employed-SR ve DEB-SR-ABC yöntemleri uygulanabilir. g10 probleminde yalnızca Employed-Onlooker-SR/rg yöntemiyle optimale yakın sonuç alınabilir. g11 ve g12 problemlerinde genel olarak tüm yöntemlerle optimal değere ulaşılmıştır. g13 probleminde PEN-ABC ($r_g = 1$), Onlooker-SR ve DEB-SR-ABC

yöntemleriyle optimale yakın sonuçlar elde edilmiştir. g14 probleminde PEN-ABC ($r_g = 50$) ve Employed-Onlooker-DE (SR) yöntemleriyle optimale en yakın sonuçlar alınmıştır, ancak hiçbir yöntem optimali bulamamıştır. g15 problemi için en iyi sonuç PEN-ABC ($r_g = 50$), Employed-Onlooker-SR ve DEB-SR-ABC yöntemleri ile alındığından çözümü için kullanılabilir. g16 probleminin çözümünde optimali bulan yöntemlerden Employed-SR, Employed-Onlooker-SR, Employed-Onlooker-SR/rg ve DEB-SR-ABC yöntemleri kullanılabilir. g17 probleminde optimale ulaşılamamıştır fakat yakın sonuçlar elde edilmiştir. Dolayısıyla PEN-ABC ($r_g = 20$) yöntemi, g17 probleminin çözümünde kullanılabilir. g18 probleminde optimale oldukça yakın sonuç bulan Employed-Onlooker-SR/rg yöntemi uygulanabilir. g19 probleminde optimal değer bulunamasa da birkaç yöntemde (PEN-ABC ($r_g = 50$), Employed-SR, Employed-Onlooker-SR, Employed-Onlooker-SR/rg) optimale yakın sonuçlar elde edilmiştir. g20 probleminde optimale en yakın sonuç Employed-Onlooker-SR/rg yöntemi ile alınmıştır. g21 probleminde Onlooker-SR yöntemiyle optimale yakın değer elde edilmiştir. g22 ve g23 problemlerinin çözümünde optimal yada optimale yakın sonuç alınamadığından başarısız olunmuştur. Son olarak g24 probleminde optimal değeri bulan Employed-SR, Onlooker-SR, Employed-Onlooker-SR, Employed-Onlooker-SR/rg ve DEB-SR-ABC yöntemleri çözüm için uygulanabilir.

4.4.6. Literatürdeki Yöntemlerle Kıyaslama

Sınırlamalı optimizasyon problemlerinin çözümü için geliştirilen üç yöntem literatürdeki diğer algoritmalarla da karşılaştırılmıştır. Tablo 4.16 ve 4.17'de problemler için elde edilen en iyi değerler ve ortalama değerler karşılaştırmalı olarak analiz edilmiştir.

4.4.6.1. PEN-ABC'nin Kıyaslanması

PEN-ABC algoritmasının r_g parametresinin farklı değerleriyle 30 koşma sonucunda bulunan en iyi sonuçları Tablo 4.16'da ve elde edilen ortalama sonuçları Tablo 4.17'de diğer algoritmalarla kıyaslamalı olarak gösterilmiştir. g01 probleminde optimal değer bulunamasa da ortalama değerler incelendiğinde GA, PSO ve DE algoritmalarından daha iyi olduğu söylenebilir. g02 probleminde optimal değere ulaşılamasa da r_g'nin tüm değerlerinde GA, PSO ve DE algoritmalarından daha iyi sonuç alınmıştır. $r_g = \{5, 10\}$

iken SR, SMES ve ABC algoritmalarından daha iyidir. Ortalama değerlere bakıldığında r_g parametresi arttıkça ortalama iyileşmiş ve diğer algorimalardan daha iyi bir değere ulaşmıştır. g03 probleminde $r_g = 50$ iken diğer r_g değerleri ilen koşulan durumlara göre en iyi sonucu almış ancak optimale ulaşamamıştır. Hem en iyi sonucuyla hem de ortalama sonucuyla sadece OPA'dan daha iyidir. PEN-ABC algoritması g04 probleminde performans diğer algoritmaların gerisinde kalmıştır. g05 probleminde SMES, PSO, DE ve ABC algoritmalarıyla en iyi sonuçlara göre benzer sonuçlar alınmıştır. Ortalama sonuçlara göre OPA ve DE algoritmalarından daha iyidir. GA'dan ise daha iyi bir performansa sahiptir. g06 probleminde en iyi sonucunu aldığı $r_g = 50$'de sadece PSO algoritmasından daha iyidir. Ortalama sonuçları kıyaslandığında g06 probleminde iyi değildir. PEN-ABC algoritması g07 probleminde diğer r_g değerleri arasında en iyi sonucunu elde ettiği $r_g = 50$ parametresiyle GA ve PSO algoritmalarından daha iyi sonuç almıştır ve ortalama sonuçlar da daha iyidir. g08 probleminde diğer algoritmalar gibi optimal değere ulaşmıştır. g09 probleminde $r_g = 20$ ve $r_g = 50$ değerleriyle optimale en yakın sonuç elde edilmiştir ve bu sonuçlarla GA'dan daha iyi, SMES ve ABC algoritması ile yakın performans göstermiştir. g10 probleminde diğer algoritmalardan daha kötü sonuçlar alınmıştır. g11 probleminde $r_g = \{20, 50\}$ iken optimal değer bulunmuş ve diğer algoritmalarla (DE algoritması hariç) benzer performans göstermiştir. Ortalama değerler incelendiğinde DE algoritmasından daha iyidir. g12 probleminde optimal değer bulunmuş ve diğer algoritmalarla benzer performans göstermiştir. g13 probleminde $r_g = 1$ parametresiyle en iyi sonucunu bulmuştur ve bu sonuçla OPA, GA, PSO, DE ve ABC algoritmalarından daha iyidir.

4.4.6.2. SR-ABC'nin Kıyaslanması

SR-ABC algoritması en iyi değerler baz alınarak Tablo 4.16 ve ortalama değer baz alınarak Tablo 4.17 ile diğer algoritmalarla kıyaslanmıştır. g01 probleminde Employed-SR yöntemi ile optimal değer bulunarak diğer algoritmalarla benzer performans elde edilmiştir. g02 probleminde en iyi sonuç Onlooker-SR ve Employed-Onlooker-SR/rg yöntemleriyle alınmıştır. Bu sonuçlar ile sadece PSO ve DE algoritmalarından performans olarak üsttedir. g03 probleminde Employed-Onlooker-SR ve Employed-Onlooker-SR/rg yöntemleriyle optimale yakın sonuçlar bularak OPA'dan

daha iyi performans göstermiştir. g04 probleminde Employed-SR, Onlooker-SR, Employed-Onlooker-SR ve Employed-Onlooker-SR/rg yöntemleriyle optimal değer yakalanmıştır. Diğer algoritmalarla (GA hariç) benzer performansa sahiptir. g05 probleminde Onlooker-SR ve Employed-Onlooker-SR/rg yöntemleriyle optimale yakın sonuçlar alınmıştır. Ancak diğer algoritmalara nazaran (GA hariç) daha alt performansa sahiptir. g06 probleminde Employed-SR ve Onlooker-SR yöntemleri ile optimal değer elde edilmiş ve diğer algoritmalarla (GA ve DE algoritmaları hariç) benzer performans göstermiştir. g07 probleminde Employed-SR ve Employed-Onlooker-SR yöntemleriyle optimale en yakın sonuçlar alınarak sadece GA'dan iyi performans göstermiştir. g08 probleminde tüm yöntemlerle optimale ulaşarak diğer algoritmalarla benzer performans elde edilmiştir. g09 probleminde Employed-SR ve Employed-Onlooker-SR/rg yöntemleriyle optimale en yakın sonuca ulaşılmıştır. Ancak diğer algoritmalar (GA haricinde) kadar başarılı değildir. g10 probleminde Employed-Onl-SR/rg yöntemiyle en iyi sonucunu almış ancak optimali yakalayamamıştır. Sadece GA'dan iyi sonuç almıştır. g11 ve g12 problemlerinde optimal değere ulaşılmıştır ve tüm yöntemlerde diğer algoritmalarla benzer performansa sahiptir. g13 probleminde ise Onlooker-SR ile en iyi sonucunu almıştır ve bu sonuçla OPA, GA, PSO, DE ve ABC algoritmalarından daha iyidir. Genel itibariyle SR-ABC algoritmasında uygulanan yöntemlerden Employed-SR, Onlooker-SR, Employed-Onlooker-SR ve Employed-Onlooker-SR/rg yöntemleri literatürdeki diğer algoritmalara benzer sonuçlar bulmuştur.

4.4.6.3. DEB-SR-ABC'nin Kıyaslanması

DEB-SR-ABC algoritması diğer algoritmalarla en iyi sonuca göre Tablo 4.16 ve ortalama sonuca göre Tablo 4.17'de karşılaştırılmıştır. DEB-SR-ABC algoritması g01 probleminde optimali bulup diğer algoritmalarla benzer performans göstermiştir. En iyi sonuca göre sadece GA'dan daha iyidir ancak ortalama sonuçlara bakıldığında GA, PSO ve DE algoritmalarından da daha iyidir. g02 probleminde optimale çok yakın sonuç bularak DE ve PSO algoritmalarından daha iyi sonuç alırken diğer algoritmalarla benzerdir. g03 probleminde optimal değer bulunup yalnız OPA'dan üst bir performans göstermiştir, diğerleriyle ise benzerdir. g04 probleminde optimal değer bulunurak diğer algoritmalarla (GA hariç) benzer performans elde edilmiştir. g05 ve g06 problemlerinde

diğer algoritmalardan düşük performans göstermiştir. g07 probleminde optimale yakın sonuç alınmıştır. GA'dan iyi, SMES, PSO ve ABC ile benzerdir. g08 probleminde diğer algoritmalarda olduğu gibi optimal değer bulunmuştur. g09 probleminde optimale çok yakın sonuç bulunmuştur ve diğer algoritmalarla (GA hariç) performans olarak yakındır. DEB-SR-ABC algoritması g10 probleminde optimalden uzak sonuç alarak diğer algoritmalardan düşük performans göstermiştir. g11 ve g12 problemlerinde optimal değer bulunmuş ve diğer algoritmalarla benzer performans göstermiştir. g13 probleminde DEB-SR-ABC algoritması optimal değeri bulamasa da OPA, GA, PSO, DE ve ABC algoritmalarından iyi performansa sahiptir.

DEB-SR-ABC algoritması SR-ABC ve PEN-ABC algoritmalarına göre ortalama değerler bakımından daha iyidir. Literatürdeki diğer algoritmalarla ortalama değer bazında performans karşılaştırması Tablo 4.18'de daha net bir şekilde gösterilmiştir. Bu tablodan yola çıkarak DEB-SR-ABC algoritmasının GA (Deb) ve DE (Deb) algoritmasından daha iyi; SR, OPA ve PSO (Deb) algoritmalarıyla ise benzer performans gösterdiği çıkarımı yapılmaktadır. Ancak ISR, SMES ve ABC (Deb) algoritmalarından daha düşük bir performansa sahiptir.

Test edilen sınırlama ele alış metotlarında Deb yönteminin ABC algoritmasının yapısına daha uygun olduğu, DEB-SR-ABC metodunun da alternatif olarak kullanılabileceği görülmüştür.

Tablo 4.16. SR, ISR, OPA, GA, SMES, PSO, DE, ABC, PEN-ABC, SR-ABC ve DEB-SR-ABC algoritmalarının 13 test fonksiyonu için elde edilen sonuçların en iyi değerleri. (-) kabul edilebilir çözümün bulunamadığı anlamına gelir.

Yöntem	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11	g12	g13
Optimal	**-15.000**	**0.803619**	**1.000**	**-30665.539**	**5126.498**	**-6961.814**	**24.306**	**0.095825**	**680.630**	**7049,25**	**0.75**	1.000	**0.053950**
SR [16]	**-15.000**	0,803515	1.000	**-30665.539**	**5126.497**	**-6961.814**	24.307	**0.095825**	**680.630**	7054,316	**0.750**	**1.000**	**0.053957**
ISR [50]	**-15.000**	**0.803619**	1.001	**-30665.539**	**5126.497**	**-6961.814**	**24.306**	**0.095825**	**680.630**	**7049.248**	**0.750**	**1.000**	0.053942
OPA [50]	**-15.000**	**0.803619**	0.747	**-30665.539**	**5126.497**	**-6961.814**	**24.306**	**0.095825**	**680.630**	**7049.248**	**0.750**	**1.000**	0.447118
GA [51]	-14.440	0.796231	0.990	-30626.053	-	-6952.472	31.097	**0.095825**	685.994	9079.770	**0.75**	**1.000**	0.134057
SMES [51]	**-15.000**	0.803601	1.000	**-30665.539**	5126.599	**-6961.814**	24.327	**0.095825**	680.632	7051.903	**0.75**	**1.000**	0.053986
PSO [52]	**-15.000**	0.669158	0.993930	**-30665.539**	5126.484	**-6961.814**	24.370153	**0.095825**	**680.630**	7049.381	0.749	**1.000**	0.085655
DE [40]	**-15.000**	0.472	1.000	**-30665.539**	5126.484	-6954.434	**24.306**	**0.095825**	**680.630**	**7049.248**	0.752	**1.000**	0.385
ABC [40]	**-15.000**	0.803598	1.000	**-30665.539**	5126.484	**-6961.814**	24.330	**0.095825**	680.634	7053.904	**0.750**	**1.000**	0.760
rg=1 (PEN-ABC	-14.414	0.802638	0.8830	-30135.576	5121.079	-7950.263	25.4577	0.095823	680.807	2101.46	0.74	**1.000**	0.05611
rg=5 (PEN-ABC)	-14.915	0.803602	0.3604	-30127.391	5129.407	-7781.586	24.7538	**0.095825**	680.653	2107.29	**0.75**	**1.000**	0.09534
rg=10 (PEN-ABC)	-14.920	0.803607	0.7294	-30241.728	5126.855	-7848.131	24.5334	**0.095825**	680.640	2114.81	**0.75**	**1.000**	0.14488
rg=20 (PEN-ABC)	-14.969	0.803591	0.8244	-30466.900	5126.534	-7464.222	24.4382	**0.095825**	680.633	2128.76	**0.75**	**1.000**	0.40804
rg=50 (PEN-ABC)	-14.986	0.803599	0.9752	-30459.850	5126.528	-6827.402	24.3659	**0.095825**	680.633	2171.19	**0.75**	**1.000**	0.56397
Employed-SR	**-15.000**	0.781265	0,9960	**-30665.539**	5133.7862	**-6961.814**	24.6819	**0.095825**	680.655	7146.44	**0.75**	**1.000**	0.27329
Onlooker-SR	-14.999	0.794766	0,9611	**-30665.539**	5127.7953	**-6961.814**	25.1593	**0.095825**	680.677	7931.90	**0.75**	**1.000**	0.06732
Emp-Onl-SR	-14.999	0.788303	1.0032	**-30665.539**	5128.492	-7950.961	24.6205	**0.095825**	680.681	7113.398	**0.75**	**1.000**	0.38624
Emp-Onl-SR/rg	-14.999	0.791442	0.9998	**-30665.539**	5127.0429	-7950.205	24.7066	**0.095825**	680.656	7062.924	**0.75**	**1.000**	0.56700
Emp-DE (SR)	-14.988	0.599782	0.5547	-30665.291	5129.6687	-7950.962	31.9284	**0.095825**	682.563	7794.773	**0.75**	**1.000**	0.50870
Onl-DE (SR)	-14.999	0.691965	0.6394	-30665.480	5139.1321	-7973	30.0470	**0.095825**	681.994	8046.979	**0.75**	**1.000**	0.46625
Emp-Onl-DE (SR)	-14.971	0.617807	0.3250	-30665.006	5167.0007	-7950.963	41.0585	**0.095825**	688.752	8269.454	**0.75**	**1.000**	0.26848
DEB-SR-ABC	**-15.000**	0.803591	**1.004**	**-30665.539**	4986.987	-7661.009	24.3438	**0.095825**	680.633	2100	**0.75**	**1.000**	0.057301

Tablo 4.17. SR, ISR, OPA, GA, SMES, PSO, DE, ABC, PEN-ABC, SR-ABC ve DEB-SR-ABC algoritmalarının 13 test fonksiyonu için elde edilen sonuçların ortalama değerleri. (-) kabul edilebilir çözümün bulunamadığı anlamına gelir.

Yöntem	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11	g12	g13
Optimal	**-15.000**	**0.803619**	**1.000**	**-30665.539**	**5126.498**	**-6961.814**	**24.306**	**0.095825**	**680.630**	**7049.25**	**0.75**	1.000	**0.053950**
SR [16]	-15.000	0.781975	1.000	-30665.539	5128.881	-6875.940	24.374	0.095825	680.656	7559.192	0.750	1.000	0.057006
ISR [50]	-15.000	0.782725	1.001	-30665.539	5126.497	-6961.814	24.306	0.095825	680.630	7049.250	0.750	1.000	0.066770
OPA [50]	-15.000	0.776283	0.257	-30665.539	5268.610	-6961.814	24.307	0.095825	680.630	7049.248	0.755	0.999889	0.964323
GA [51]	-14.236	0.788588	0.976	-30590.455	-	-6872.204	34.980	0.095799	692.064	10003.225	0.75	1.000	-
SMES [51]	-15.000	0.785238	1.000	-30665.539	5174.492	-6961.284	24.475	0.095825	680.643	7253.047	0.75	1.000	0.166385
PSO [52]	-14.710	0.419960	0.764813	-30665.539	5135.973	-6961.814	32.407	0.095825	680.630	7205.5	0.749	0.998875	0.569358
DE [40]	-14.555	0.665	1.000	-30665.539	5264.270	-	24.310	0.095825	680.630	7147.334	0.901	1.000	0.872
ABC [40]	-15.000	0.792412	1.000	-30665.539	5185.714	-6961.813	24.473	0.095825	680.640	7224.407	0.750	1.000	0.968
rg=1 (PEN-ABC	-12.120	0.787720	0.0082	-29267.342	5189.360	-5840.023	31.673	0.094926	680.915	11098.665	0.927	1.000	0.45962
rg=5 (PEN-ABC)	-14.604	0.789204	0.0240	-29210.471	5305.625	-4984.099	25.546	0.095629	680.693	8166.607	0.852	1.000	0.79161
rg=10 (PEN-ABC)	-14.794	0.791964	0.1141	-29388.250	5312.793	-5515.717	25.007	0.095783	680.655	9386.364	0.783	1.000	0.75960
rg=20 (PEN-ABC)	-14.858	0.786921	0.2189	-29385.895	5201.197	-5149.604	24.767	0.095799	680.648	8098.925	0.749	1.000	0.83561
rg=50 (PEN-ABC)	-14.948	0.793056	0.6737	-29337.438	5316.717	-5523.459	24.591	0.095629	680.642	9044.092	0.749	1.000	0.88431
Employed-SR	-13.766	0.721954	0,7576	-30635.714	5495.967	-6961.814	29.3491	0.095825	680.755	7820.773	0.749	1.000	1.24207
Onlooker-SR	-13.099	0.717791	0,6143	-30665.539	5913.843	-6961.814	27.2965	0.095825	680.958	10464.995	0.753	1.000	0.69821
Emp-Onl-SR	-12.679	0.623135	0.4954	-30653.502	5451.812	-6992.820	57.0992	0.095825	681.416	10395.296	0.832	0.999	0.92376
Emp-Onl-SR/rg	-11.916	0.621560	0.6898	-30662.664	5616.404	-7027.668	98.4280	0.095825	681.630	10620.716	0.776	0.999	1.04602
Emp-DE (SR)	-10.399	0.466272	0.0525	-30632.953	5580.917	-6891.767	315.393	0.095824	1017.019	12152.608	0.840	0.999	1.50786
Onl-DE (SR)	-11.631	0.487656	0.0734	-30588.714	5656.963	-6980.580	483.8817	0.095825	1014.704	11304.605	0.824	0.999	1.73945
Emp-Onl-DE (SR)	-10.719	0.432700	0.0108	-30661.255	5701.441	-6440.205	376.6125	0.095817	61033.509	14177.442	0.832	0.999	1.21924
DEB-SR-ABC	-15.000	0.791671	1.002	-30665.539	5559.641	-6344.100	24.513	0.095825	680.641	11091.682	0.75	1.000	0.59645

Tablo 4.18. DEB-SR-ABC algoritmasının diğer algoritmalarla karşılaştırıldığında başarı oranı. + algoritmanın daha iyi, - daha kötü olduğu anlamına gelir. Benzer performans gösterenlerin her ikisi de + alınmıştır.

Prob.	DEB-SR-ABC-SR		DEB-SR-ABC-ISR		DEB-SR-ABC-OPA		DEB-SR-ABC-GA		DEB-SR-ABC-SMES		DEB-SR-ABC-PSO		DEB-SR-ABC-DE		DEB-SR-ABC - ABC	
	DEB-SR-ABC	SR	DEB-SR-ABC	ISR	DEB-SR-ABC	OPA	DEB-SR-ABC	GA	DEB-SR-ABC	SMES	DEB-SR-ABC	PSO	DEB-SR-ABC	DE	DEB-SR-ABC	ABC
g01	+	+	+	+	+	+	+	-	+	+	+	-	+	-	+	+
g02	+	-	+	-	+	-	+	-	+	-	+	-	+	-	-	+
g03	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
g04	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+	+
g05	-	+	-	+	-	-	-	-	-	+	-	+	-	+	-	+
g06	-	+	-	+	-	+	-	+	-	+	-	+	+	-	-	+
g07	-	+	-	+	-	+	+	-	-	+	+	-	-	+	-	+
g08	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+	+
g09	+	-	-	+	-	+	+	-	+	-	-	+	-	+	-	+
g10	-	-	-	+	-	+	-	-	-	+	-	+	-	+	-	+
g11	+	+	+	+	+	-	+	+	+	+	+	-	+	-	+	+
g12	+	+	+	+	+	-	+	+	+	+	+	-	+	+	+	+
g13	-	+	-	+	+	-	+	-	-	+	-	+	+	-	+	-
Toplam	8	9	7	11	8	7	10	3	8	10	8	7	9	7	7	11

5. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

5.1. Tartışma, Sonuç ve Öneriler

Bu tez çalışmasında sınırlamalı optimizasyon problemlerini çözmek amacıyla üç yöntem (PEN-ABC, SR-ABC ve DEB-SR-ABC) önerilmiştir. Standart ABC algoritmasına bu yöntemler entegre edilerek 24 test problemi üzerindeki performansı incelenmiştir. Sonuçlar incelendiğinde sınırlamalı optimizasyon problemlerinin çözümü belli paramatre değerleriyle sağlanmıştır. PEN-ABC algoritmasında r_g parametresinin artan değerleriyle sonucun iyileştiği görülmüştür. Stokastik beslemeli ABC algoritmasında (SR-ABC) stokastik yapı ABC algoritmasının işçi ve gözcü arı fazlarına entegre edilerek sonuçlar incelendiğinde dört yöntemle de benzer performans elde edilmiştir. Ancak stokastik sıralamanın tek başına ABC algoritmasının yapısına uygun olmadğı görülmüştür. Ayrıca DE algoritmasının mutant vektör oluşturma denkleminin SR-ABC algoritmasının komşu üretme mekanizmasında kullanımında sadece gözcü arı fazına entegre edildiğinde daha iyi sonuç alınmıştır. DEB-SR-ABC algoritmasında Deb'in kurallarına ek olarak stokastik yapı kullanılarak popülasyona çeşitlilik katılması amaçlanmış ancak çeşitlilik katıldığında Deb'in kurallarının temel halinin kullanıldığı yöntem kadar başarılı olamamıştır. Dolayısıyla Deb yönteminin ABC algoritmasının yapısına daha uygun olduğu, DEB-SR metodunun da alternatif olarak kullanılabileceği görülmüştür. Önerilen yöntemler literatürdeki diğer optimizasyon algoritmalarıyla, literatürde sıklıkla kullanılan 13 test problemi üzerinden karşılaştırılmıştır. Yapılan istatistiksel sonuçlar incelendiğinde bazı problemlerde daha iyi sonuç alındığı ancak genel olarak bakıldığında ise diğer algoritmalara yakın performans gösterdiği söylenebilir.

KAYNAKLAR

1. Akay, B., 2009. Nümerik optimizasyon problemlerinde yapay arı kolonisi (artificial bee colony) algoritmasının performans analizi, Doktora Tezi, Erciyes Üniversitesi.

2. Arora, J.S., 2004. Introduction to optimum design, Elsevier.

3. Rao, S.S., 2009. Engineering optimization: theory and practice, John Wiley & Sons.

4. Vesterstrom, J. and Riget, J., 2002. Particle swarm extensions for improved local, multimodal and dynamic searc in numerical optimization, Yüksek Lisans Tezi, Univ Aarhus, Denmark.

5. Qin, A.K., Huang, V.L., and Suganthan, P.N., 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization, **IEEE transactions on Evolutionary Computation**, **13**(2):398–417.

6. Karaboga, D. and Akay, B., 2009. A comparative study of artificial bee colony algorithm, **Applied mathematics and computation**, **214**(1):108–132.

7. Parsopoulos, K.E. and Vrahatis, M.N., 2002. Particle swarm optimization method for constrained optimization problems, **Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies**, **76**(1):214–220.

8. Sheth, P. and Umbarkar, A., 2015. Constrained Optimization Problems Solving Using Evolutionary Algorithms: A Review, *Computational Intelligence and Communication Networks (CICN), 2015 International Conference on*, 1251–1257, IEEE.

9. Huang, Z., Wang, C., and Tian, H., 2009. A genetic algorithm with constrained sorting method for constrained optimization problems, *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1, 806–811, IEEE.

10. Bacanin, N. and Tuba, M., 2012. Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators, **Studies in Informatics and Control**, **21**(2):137–146.

11. Lu, H. and Chen, W., 2008. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems, **Journal of Global Optimization**, **41**(3):427–445.

12. Michalewicz, Z. and Schoenauer, M., 1996. Evolutionary algorithms for constrained parameter optimization problems, **Evolutionary computation**, **4**(1):1–32.

13. Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, **Computer methods in applied mechanics and engineering**, **191**(11-12):1245–1287.

14. Joines, J.A. and Houck, C.R., 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's, *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 579–584, IEEE.

15. Siedlecki, W. and Sklansky, J., 1993. Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition, Handbook of pattern recognition and computer vision, 108–123, World Scientific.

16. Runarsson, T.P. and Yao, X., 2000. Stochastic ranking for constrained evolutionary optimization, **IEEE Transactions on evolutionary computation**, **4**(3):284–294.

17. Deb, K., 2000. An efficient constraint handling method for genetic algorithms, **Computer methods in applied mechanics and engineering**, **186**(2-4):311–338.

18. Wu, G., Pedrycz, W., Suganthan, P.N., and Mallipeddi, R., 2015. A variable reduction strategy for evolutionary algorithms handling equality constraints, **Applied Soft Computing**, **37**:774–786.

19. Abo-Bakr, R.M. and Mujeed, T.A., 2015. Solving nonlinear constrained optimization problems using hybrid evolutionary algorithms, *Computer Engineering Conference (ICENCO), 2015 11th International*, 150–156, IEEE.

20. Mezura-Montes, E. and Coello, C.A.C., 2003. Adding a diversity mechanism to a simple evolution strategy to solve constrained optimization problems, *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, 6–13, IEEE.

21. Liu, H., Cai, Z., and Wang, Y., 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, **Applied Soft Computing**, **10**(2):629–640.

22. Wang, Y. and Cai, Z., 2011. Constrained evolutionary optimization by means of (μ+ λ)-differential evolution and improved adaptive trade-off model, **Evolutionary Computation**, **19**(2):249–285.

23. Mallipeddi, R. and Suganthan, P.N., 2010. Ensemble of constraint handling techniques, **IEEE Transactions on Evolutionary Computation**, **14**(4):561–579.

24. Mezura-Montes, E., Miranda-Varela, M.E., and del Carmen Gómez-Ramón, R., 2010. Differential evolution in constrained numerical optimization: an empirical study, **Information Sciences**, **180**(22):4223–4262.

25. Sardar, S., Maity, S., Das, S., and Suganthan, P.N., 2011. Constrained real parameter optimization with a gradient repair based differential evolution algorithm, *Differential Evolution (SDE), 2011 IEEE Symposium on*, 1–8, IEEE.

26. Mohamed, A.W. and Sabry, H.Z., 2012. Constrained optimization based on modified differential evolution algorithm, **Information Sciences**, **194**:171–208.

27. Jia, G., Wang, Y., Cai, Z., and Jin, Y., 2013. An improved (μ+ λ)-constrained differential evolution for constrained optimization, **Information Sciences**, **222**:302–322.

28. Gao, W., Yen, G.G., and Liu, S., 2015. A dual-population differential evolution with coevolution for constrained optimization., **IEEE Trans. Cybernetics**, **45**(5):1094–1107.

29. Zheng, J., Wu, Q., and Song, W., 2007. An improved particle swarm algorithm for solving nonlinear constrained optimization problems, *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, 112–117, IEEE.

30. Sun, C.l., Zeng, J.c., and Pan, J.S., 2009. A new vector particle swarm optimization for constrained optimization problems, *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 1, 485–488, IEEE.

31. Miao, K., Li, L., Yang, X.L., and Huo, Y.Y., 2009. Two-stage Probing Method for Constrained Optimization Problems through Particle Swarm Optimization, *Information Science and Engineering (ICISE), 2009 1st International Conference on*, 3894–3897, IEEE.

32. Sun, C., Zeng, J., Chu, S., Roddick, J.F., and Pan, J., 2011. Solving constrained optimization problems by an improved particle swarm optimization, *Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on*, 124–128, IEEE.

33. Wei, J. and Zhang, M., 2011. A memetic particle swarm optimization for constrained multi-objective optimization problems, *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 1636–1643, IEEE.

34. Li, L.D., Yu, X., Li, X., and Guo, W., 2011. A dynamic neighbourhood particle swarm optimisation algorithm for constrained optimisation, *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, 3042–3047, IEEE.

35. Liu, Z. and Hui, Q., 2012. A constraint-handling technique for particle swarm optimization, *World Automation Congress (WAC), 2012*, 1–6, IEEE.

36. Elsayed, S.M., Sarker, R.A., and Mezura-Montes, E., 2013. Particle swarm optimizer for constrained optimization, *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2703–2711, IEEE.

37. Wei, J. and Jia, L., 2013. A novel particle swarm optimization algorithm with local search for dynamic constrained multi-objective optimization problems, *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2436–2443, IEEE.

38. Wu, J.Y., 2015. An improved quantum-behaved particle swarm optimization method for solving constrained global optimization problems, *Communications and Information Technologies (ISCIT), 2015 15th International Symposium on*, 157–160, IEEE.

39. Karaboga, D. and Basturk, B., 2007. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, *International fuzzy systems association world congress*, 789–798, Springer.

40. Karaboga, D. and Akay, B., 2011. A modified artificial bee colony (ABC) algorithm for constrained optimization problems, **Applied soft computing**, **11**(3):3021–3031.

41. Mezura-Montes, E. and Cetina-Domínguez, O., 2012. Empirical analysis of a modified artificial bee colony for constrained numerical optimization, **Applied Mathematics and Computation**, **218**(22):10943–10973.

42. Brajevic, I. and Tuba, M., 2013. An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems, **Journal of Intelligent Manufacturing**, **24**(4):729–740.

43. Aguilar-Justo, A.E., Mezura-Montes, E., and Coello, C.A.C., 2014. Memetic Modified Artificial Bee Colony for constrained optimization, *Power, Electronics and Computing (ROPEC), 2014 IEEE International Autumn Meeting on*, 1–6, IEEE.

44. Li, X. and Yin, M., 2014. Self-adaptive constrained artificial bee colony for constrained numerical optimization, **Neural Computing and Applications**, **24**(3-4):723–734.

45. Brajevic, I., 2015. Crossover-based artificial bee colony algorithm for constrained optimization problems, **Neural Computing and Applications**, **26**(7):1587–1601.

46. Akay, B. and Karaboga, D., 2017. Artificial bee colony algorithm variants on constrained optimization, **Journal of Optimization and Control: Theories & Applications**, **7**(1):98–111.

47. Karaboga, D. and Akay, B., 2009. A survey: algorithms simulating bee swarm intelligence, **Artificial intelligence review**, **31**(1-4):61.

48. Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization, Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

49. Tereshko, V., 2000. Reaction-diffusion model of a honeybee colony's foraging behaviour, *International Conference on Parallel Problem Solving from Nature*, 807–816, Springer.

50. Runarsson, T.P. and Yao, X., 2005. Search biases in constrained evolutionary optimization, **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, **35**(2):233–243.

51. Mezura-Montes, E. and Coello, C.A.C., 2005. A simple multimembered evolution strategy to solve constrained optimization problems, **IEEE Transactions on Evolutionary computation**, **9**(1):1–17.

52. Munoz Zavala, A.E., Aguirre, A.H., and Villa Diharce, E.R., 2005. Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 209–216, ACM.

53. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.C., and Deb, K., 2006. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, **Journal of Applied Mechanics**, **41**(8):8–31.

EK-1

KULLANILAN TEST PROBLEMLERİ

1.1. Sınırlamalı Test Problemleri

g01:

Minimize

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Subject to:

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

$0 \leq x_i \leq 1$ $(i = 1, ..., 9, 13)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$. global optimum: $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, $f(x^*) = -15$. g_1, g_2, g_3, g_7, g_8 ve g_9 aktif sınırlamalardır.

g02:

Maksimize

$$f(\vec{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \right|$$

Subject to:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

$n = 20$ ve $0 \leq x_i \leq 10$ $(i = 1, ..., n)$. Bulunan en iyi çözüm $f(x^*) = 0.80361910412559$. g_1 sınırlaması aktifliğe yakındır.

g03:

Maksimize

$$f(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

Subject to:

$$h_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

$n = 10$ ve $0 \leq x_i \leq 1$ $(i = 1, ..., n)$. $x_i^* = 1/\sqrt{n}$'de global optimum $f(x^*) = 1.00050010001000$.

g04:

Minimize

$$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

Subject to:

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_2x_4 - 92 \leq 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3 * x_5 \leq 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. $x^* = (78, 33, 29.9952560256815985, 45, 36.7758129057882073)$, global optimum $f(x^*) = -30665.539$. g_1 ve g_6 sınırlamaları aktiftir.

g05:

Minimize

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

Subject to:

$$g_1(\vec{x}) = -x_4 - x_3 - 0.55 \leq 0$$

$$g_2(\vec{x}) = -x_3 - x_4 - 0.55 \leq 0$$

$$h_3(\vec{x}) = 1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\vec{x}) = 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\vec{x}) = 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

$0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ ve $-0.55 \leq x_4 \leq 0.55$. $x^* = (679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.396233485215178266)$, $f(x^*) = 5126.4967140071$

g06:

Minimize

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Subject to:

$$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

$13 \leq x_1 \leq 100$ ve $0 \leq x_2 \leq 100$. $x^* = (14.09500000000000064, 0.8429607892154795668)$ ve $f(x^*) = -6961.81387558015$. Her iki sınırlama da aktif.

g07:

Minimize

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Subject to:

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0$$
$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_10 \leq 0$$

$-10 \leq x_i \leq 10$ $(i = 1, ..., 10)$. Global optimum $x^* = (2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347)$, $f(x^*) = 24.30620906818$. g_1, g_2, g_3, g_4, g_5 ve g_6 sınırlamaları aktif.

g08:

Maksimize

$$f(\vec{x}) = \frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to:

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

$0 \leq x_1 \leq 10$ ve $0 \leq x_2 \leq 10$. Global optimum $x^* = (1.22797135260752599, 4.24537336612274885)$, $f(x^*) = 0.0958250414180359$.

g09:

Minimize

$$f(\vec{x}) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4+4x_6x_7+10x_6-8x_7$$

Subject to:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 + 8x_7 \leq 0$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

$-10 \leq x_i \leq 10$ $(i = 1, ..., 7)$. Global optimum $x^* = (2.33049935147405174, 1.95137236847114592, 0.477541399510615805, 4.36572624923625874, 0.624486959100388983, 1.03813099410962173, 1.5942266780671519)$, $f(x^*) = 680.630057374402$.

g10:

Minimize

$$f(\vec{x}) = x_1 + x_2 + x_3$$

Subject to:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

$100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000$, $(i = 2, 3)$, $10 \leq x_i \leq 1000$, $(i = 4, ..., 8)$. Global optimum $x^* = (579.306685017979589, 1359.97067807935605, 5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852, 395.601173702746735)$, $f(x^*) = 7049.24802052867$. g_1, g_2 ve g_3 sınırlamaları aktif.

g11:

Minimize

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2$$

Subject to:

$$h(\vec{x}) = x_2 - x_1^2 = 0$$

$-1 \leq x_1 \leq 1,\ -1 \leq x_2 \leq 1$. Global optimum $x^* = (\pm 1/\sqrt{2}, 1/2)$, $f(x^*) = 0.7499$.

g12:

Maksimize

$$f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$$

Subject to:

$$g_1(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

$0 \leq x_i \leq 10\ (i = 1, 2, 3)$ ve $p, q, r = 1, 2, ..., 9$. Global optimum $x^* = (5, 5, 5)$, $f(x^*) = 1$.

g13:

Minimize

$$f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$

Subject to:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 0$$

$$h_2(\vec{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

$-2.3 \leq x_i \leq 2.3\ (i = 1, 2),\ -3.2 \leq x_i \leq 3.2,\ (i = 3, 4, 5)$. Global optimum $x^* = (-1.717143, 1.5957091, -0.736413, -0.763645)$, $f(x^*) = 0.0539498$.

g14:

Minimize

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

Subject to:

$$h_1(\vec{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_1 0 - 2 = 0$$

$$h_2(\vec{x}) = x_4 + 2x_5 + x_6 x_7 - 1 = 0$$

$$h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_1 0 - 1 = 0$$

$0 < x_i \leq 10,\ (i = 1, ..., 10)$ ve $c_1 = -6.089,\ c_2 = -17.164,\ c_3 = -34.054,\ c_4 = -5.914,\ c_5 = -24.721,\ c_6 = -14.986,\ c_7 = -24.1,\ c_8 = -10.708,\ c_9 = -26.662,\ c_1 0 = -22.179$. Global optimum $x^* =$

$(0.0406684113216282, 0.147721240492452, 0.783205732104114,$
$0.00141433931889084, 0.485293636780388, 0.000693183051556082,$
$0.0274052040687766, 0.0179509660214818, 0.0373268186859717,$
$0.0968844604336845)$, $f(x^*) = -47.7648884594915$.

g15:

Minimize

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

Subject to:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$
$$h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$0 < x_i \leq 10, (i = 1, 2, 3)$. Global optimum $x^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$, $f(x^*) = 961.715022289961$.

g16:

Minimize

$$f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_16 + 0.0321y_12 + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

Subject to:

$$g_1(\vec{x}) = \frac{0.25}{0.72} y_5 - y_4 \leq 0$$

$$g_2(\vec{x}) = x_3 - 1.5x_2 \leq 0$$

$$g_3(\vec{x}) = 3496 \frac{y_2}{c_{12}} - 21 \leq 0$$

$$g_4(\vec{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$$

$$g_5(\vec{x}) = 213.1 - y_1 \leq 0$$

$$g_6(\vec{x}) = y_1 - 405.23 \leq 0$$

$$g_7(\vec{x}) = 17.505 - y_2 \leq 0$$

$$g_8(\vec{x}) = y_2 - 1052.6667 \leq 0$$

$$g_9(\vec{x}) = 11.275 - y_3 \leq 0$$

$$g_{10}(\vec{x}) = y_3 - 35.03 \leq 0$$

$$g_{11}(\vec{x}) = 214.228 - y_4 \leq 0$$

$$g_{12}(\vec{x}) = y_4 - 665.585 \leq 0$$

$$g_{13}(\vec{x}) = 7.458 - y_5 \leq 0$$

$$g_{14}(\vec{x}) = y_5 - 584.463 \leq 0$$

$$g_{15}(\vec{x}) = 0.961 - y_6 \leq 0$$

$$g_{16}(\vec{x}) = y_6 - 265.916 \leq 0$$

$$g_{17}(\vec{x}) = 1.612 - y_7 \leq 0$$

$$g_{18}(\vec{x}) = y_7 - 7.046 \leq 0$$

$$g_{19}(\vec{x}) = 0.146 - y_8 \leq 0$$

$$g_{20}(\vec{x}) = y_8 - 0.222 \leq 0$$

$$g_{21}(\vec{x}) = 107.99 - y_9 \leq 0$$

$$g_{22}(\vec{x}) = y_9 - 273.366 \leq 0$$

$$g_{23}(\vec{x}) = 922.693 - y_10 \leq 0$$

$$g_{24}(\vec{x}) = y_10 - 1286.105 \leq 0$$

$$g_{25}(\vec{x}) = 926.832 - y_11 \leq 0$$

$$g_{26}(\vec{x}) = y_1 1 - 1444.046 \leq 0$$

$$g_{27}(\vec{x}) = 18.766 - y_1 2 \leq 0$$

$$g_{28}(\vec{x}) = y_1 2 - 537.141 \leq 0$$

$$g_{29}(\vec{x}) = 1072.163 - y_1 3 \leq 0$$

$$g_{30}(\vec{x}) = y_1 3 - 3247.039 \leq 0$$

$$g_{31}(\vec{x}) = 8961.448 - y_1 4 \leq 0$$

$$g_{32}(\vec{x}) = y_1 4 - 26844.086 \leq 0$$

$$g_{33}(\vec{x}) = 0.063 - y_1 5 \leq 0$$

$$g_{34}(\vec{x}) = y_1 5 - 0.386 \leq 0$$

$$g_{35}(\vec{x}) = 71084.33 - y_1 6 \leq 0$$

$$g_{36}(\vec{x}) = -140000 + y_1 6 \leq 0$$

$$g_{37}(\vec{x}) = 2802713 - y_1 7 \leq 0$$

$$g_{38}(\vec{x}) = y_1 7 - 12146108 \leq 0$$

$$
\begin{aligned}
y_1 &= x_2 + x_3 + 41.6 \\
c_1 &= 0.024x_4 - 4.62 \\
y_2 &= \frac{12.5}{c_1} + 12 \\
c_2 &= 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 &= 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 &= \frac{c_2}{c_3} \\
y_4 &= 19y_3 \\
c_4 &= 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
c_5 &= 100x_2 \\
c_6 &= x_1 - y_3 - y_4 \\
c_7 &= 0.950 - \frac{c_4}{c_5} \\
y_5 &= c_6c_7 \\
y_6 &= x_1 - y_5 - y_4 - y_3 \\
c_8 &= (y_5 + y_4)0.995 \\
y_7 &= \frac{c_8}{y_1} \\
y_8 &= \frac{c_8}{3798} \\
c_9 &= y_7 - \frac{0.0663y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321y_1 \\
y_{10} &= 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
y_{11} &= 1.71x_1 - 0.452y_4 + 0.58y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75y_2)(0.995x_1) \\
c_{12} &= 0.995y_10 + 1998 \\
y_{12} &= c_{10}x_1 + \frac{c_{10}}{c_{12}} \\
y_{13} &= c_{12} - 1.75y_2 \\
y_{14} &= 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5} \\
c_{13} &= 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \\
y_{15} &= \frac{y_{13}}{c_{12}} \\
y_{16} &= 148000 - 331000y_{15} + 40y13 - 61y_{15}y_{13} \\
c_{14} &= 2324y_{10} - 28740000y_2 \\
y_{17} &= 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} &= \frac{y_{13}}{y15} - \frac{y_{13}}{0.52} \\
c_{16} &= 1.104 - 0.72y_{15} \\
c_{17} &= y_8 + x_5
\end{aligned}
$$

ve $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$, $25 \leq x_5 \leq 84.1988$. Global optimum $x^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$, $f(x^*) = -1.90515525853479$.

g17:

Minimize

$$f(\vec{x}) = f(x_1) + f(x_2)$$

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

Subject to:

$$h_1(\vec{x}) = -x_1 + 300 - \frac{x_3 x_4}{131.078} cos(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} cos(1.47588) = 0$$
$$h_2(\vec{x}) = -x_2 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} cos(1.47588) = 0$$
$$h_3(\vec{x}) = -x_5 - \frac{x_3 x_4}{131.078} sin(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} sin(1.47588) = 0$$
$$h_4(\vec{x}) = 200 - \frac{x_3 x_4}{131.078} sin(1.48477 - x_6) + \frac{0.90798 x_4^2}{131.078} sin(1.47588) = 0$$

$0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ ve $0 \leq x_6 \leq 0.5236$. Global optimum $x^* = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.07314823120842871 28)$, $f(x^*) = 8853.53967480648$.

g18:

Minimize

$$f(\vec{x}) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$$

Subject to:

$$g_1(\vec{x}) = x_3^2 + x_4^2 - 1 \leq 0$$
$$g_2(\vec{x}) = x_9^2 - 1 \leq 0$$
$$g_3(\vec{x}) = x_5^2 + x_6^2 - 1 \leq 0$$
$$g_4(\vec{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$$
$$g_5(\vec{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$
$$g_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$
$$g_7(\vec{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$$
$$g_8(\vec{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$$
$$g_9(\vec{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$
$$g_{10}(\vec{x}) = x_2 x_3 - x_1 x_4 \leq 0$$
$$g_{11}(\vec{x}) = -x_3 x_9 \leq 0$$
$$g_{12}(\vec{x}) = x_5 x_9 \leq 0$$
$$g_{13}(\vec{x}) = x_6 x_7 - x_5 x_8 \leq 0$$

$-10 \leq x_i \leq 10$, $(i = 1, ..., 8)$ ve $0 \leq x_9 \leq 20$. Global optimum $x^* = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$, $f(x^*) = -0.866025403784439$.

g19:

Minimize

$$f(\vec{x}) = \sum_{j=1}^{5} \sum_{i=1}^{5} c_{ij} x_{(10+i)} x_{(10+j)} + 2 \sum_{j=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i$$

Subject to:

$$g_j(x) = -2 \sum_{i=1}^{5} c_{ij} x_{(10+i)} - e_j + \sum_{i=1}^{10} a_{ij} x_i, \; j = 1, ..., 5$$

$\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$, $0 \leq x_i \leq 10$, $i = (1, ..., 15)$ ve kalan diğer değerler Tablo EK-1.1'de verilmiştir. Global optimum $x^* = (1.66991341326291344e^1 7, 3.95378229282456509e^1 6, 3.94599045143233784, 1.06036597479721211e^1 6, 3.2831773458454161, 9.99999999999999822,$

$1.12829414671605333e^17, 1.2026194599794709e^17, 2.50706276000769697e^15,$
$2.24624122987970677e^15, 0.370764847417013987, 0.278456024942955571,$
$0.523838487672241171, 0.388620152510322781, 0.298156764974678579), \quad f(x^*) = 32.6555929502463.$

g20:

Minimize

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$$

Subject to:

$$g_i(\vec{x}) = \frac{x_i + x_{(i+12)}}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \ i = 1, 2, 3$$
$$g_i(\vec{x}) = \frac{(x_{i+3} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \ i = 4, 5, 6$$
$$h_1(\vec{x}) = \frac{x_{(i+12)}}{b_{i+12} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, \ i = 1, ..., 12$$
$$h_{13}(\vec{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$
$$h_{14}(\vec{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

$k = (0.7302)(530)(\frac{14.7}{40})$ ve kalan değerler Tablo EK-1.2'de verilmiştir. Optimum çözüm şimdiye kadar bulunamamıştır.

g21:

Minimize

$$f(\vec{x}) = x_1$$

Subject to:

$$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$
$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$
$$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$
$$h_3(\vec{x}) = -x_5 + ln(-x_4 + 900) = 0$$
$$h_4(\vec{x}) = -x_6 + ln(x_4 + 300) = 0$$
$$h_5(\vec{x}) = -x_7 + ln(-2x_4 + 700) = 0$$

$0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ ve $4.5 \leq x_7 \leq 6.25$. Global optimum $x^* = (193.724510070034967, 5.56944131553368433e^{-27}, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$, $f(x^*) = 193.724510070035$.

g22:

Minimize

$$f(\vec{x}) = x_1$$

Subject to:

$$g_1(\vec{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} x_4^{0.6} \leq 0$$

$$h_1(\vec{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$$

$$h_2(\vec{x}) = x_6 - 100000x_8 - 100000x_9 = 0$$

$$h_3(\vec{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$

$$h_4(\vec{x}) = x_5 + 100000x_10 - 3.3 \times 10^7 = 0$$

$$h_5(\vec{x}) = x_6 + 100000x_11 - 4.4 \times 10^7 = 0$$

$$h_6(\vec{x}) = x_7 + 100000x_12 - 6.6 \times 10^7 = 0$$

$$h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0$$

$$h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0$$

$$h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0$$

$$h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0$$

$$h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0$$

$$h_{12}(\vec{x}) = -x_{18} + ln(x_{10} - 100) = 0$$

$$h_{13}(\vec{x}) = -x_{19} + ln(-x_8 - 300) = 0$$

$$h_{14}(\vec{x}) = -x_{20} + ln(x_{16}) = 0$$

$$h_{15}(\vec{x}) = -x_{21} + ln(-x_9 + 400) = 0$$

$$h_{16}(\vec{x}) = -x_{22} + ln(x_{17}) = 0$$

$$h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$

$$h_{18}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$

$$h_{19}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} - x_{15}x_{22} + 100 = 0$$

$0 \leq x_1 \leq 20000, 0 \leq x_2, x_3, x_4 \leq 1 \times 10^6, 0 \leq x_5, x_6, x_7 \leq 4 \times 10^7, 100 \leq x_8 \leq 299.99, 100 \leq x_9 \leq 399.99, 100.01 \leq x_{10} \leq 300, 100 \leq x_{11} \leq 400, 100 \leq x_{12} \leq 600, 0 \leq x_{13}, x_{14}, x_{15} \leq 500, 0.01 \leq x_{16} \leq 300, 0.01 \leq x_{17} \leq 400, -4.7 \leq x_{18}$ ve $x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. Global optimum $x^* = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395)$, $f(x^*) = 236.430975504001$.

g23:

Minimize

$$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$$

Subject to:

$$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$$

$$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$$

$0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ ve $0.01 \leq x_9 \leq 0.03$. Global optimum $x^* = (0.0051000000000259465, 99.9947000000000514, 9.01920162996045897e^{-18}, 99.9999000000000535, 0.00010000000027086086, 2.75700683389584542e^{-14}, 99.9999999999999574, 2000.0100000100000100008)$,
$f(x^*) = -400.055099999999584$.

g24:

Minimize

$$f(\vec{x}) = -x_1 - x_2$$

Subject to:

$$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

$0 \leq x_1 \leq 3$ ve $0 \leq x_2 \leq 4$. Global optimum $x^* = (2.329520197477623, 17849307411774)$, $f(x^*) = -5.50801327159536$.

Tablo EK-1.1. g19 test problemi için veri seti.

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	-10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0.4	2
a_{3j}	-3.5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

Tablo EK-1.2. g20 test problemi için veri seti.

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı	:	Demet ALICI KARACA
Uyruğu	:	Türkiye (T.C.)
Doğum Tarihi ve Yeri	:	1991 - SİVAS
Telefon	:	0 352 207 66 66
E-posta	:	demet.karaca@erciyes.edu.tr
Adres	:	Erciyes Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölüm Başkanlığı 38039, Melikgazi KAYSERİ TÜRKİYE

EĞİTİM

Derece	Kurum	Mez.Yılı
Lisans	ERÜ Müh. Fakültesi Bilgisayar Müh., KAYSERİ	2015

İŞ DENEYİMİ

Yıl	Kurum	Görev
2016-Halen	Erzincan Binali Yıldırım Üniversitesi, Müh. Fakültesi, Bilgisayar Müh., ERZİNCAN	Araştırma Görevlisi